IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX LLC and AUSTIN STORMS, Plaintiffs, v. LANCIUM LLC, MICHAEL T. MCNAMARA, and RAYMOND E. CLINE, JR. Defendants.	)	**REDACTED PUBLIC VERSION** C.A. No. 21-534-MN-CJB

DECLARATION OF CHELSEA MURRAY IN SUPPORT OF PLAINTIFFS' BRIEF IN OPPOSITION TO DEFENDANTS' FIRST MOTION FOR SUMMARY JUDGMENT

I, Chelsea Murray, declare as follows:

1. I am one of the attorneys representing the Plaintiffs Bearbox LLC and Austin Storms in this matter. I am an attorney admitted to practice law in the State of Illinois and before this Court (pro hac vice), and I am a associate of the law firm of Marshall, Gerstein & Borun LLP.

2. I am above the age of eighteen. I make this declaration of my own personal knowledge. If called to testify as to the truth of the matters stated herein, I could and would testify competently.

3. I submit this declaration and the attached exhibits in support of Plaintiffs' Brief in Opposition to Defendants' First Motion For Summary Judgment.

4. Attached hereto as Exhibit A are true and correct copies of BB00000001-BB00000089, produced by Plaintiffs in this case.

5. Attached hereto as Exhibit B are true and correct copies of BB00000090 – BB00000097 (email and attachments from A. Storms to M. McNamara dated May 9, 2019), produced by Plaintiffs in this case.

6. Attached hereto as Exhibit C are true and correct copies of excerpts from Nikolaus Baer's Expert Report of Source Code (and Appendix G).

7. Attached hereto as Exhibit D are true and correct copies of LANCIUM00015068-LANCIUM00015072, produced by Defendants in this case.

8. Attached hereto as Exhibit E are true and correct copies of excerpts from the Deposition Transcript of Austin Storms.

9. Attached hereto as Exhibit F are true and correct copies of excerpts from the Deposition Transcript of Michael McNamara.

10. Attached hereto as Exhibit G are true and correct copies of BB10004960 (text message exchange between Storms/Bearbox and McNamara), produced by Plaintiffs in this case.

11. Attached hereto as Exhibit H are true and correct copies of excerpts from the Deposition Transcript of Raymond Cline.

12. Attached hereto as Exhibit I are true and correct copies of excerpts from US Patent Application No. WO2019/139632.

13. Attached hereto as Exhibit J are true and correct copies of excerpts from the Expert Report of Mark Ehsani, Ph.D.

14. Attached hereto as Exhibit K are true and correct copies of excerpts from the Deposition Transcript of Mark Ehsani.

15. Attached hereto as Exhibit L are true and correct copies of LANCIUM00014645-52, produced by Defendants in this case.

16. Attached hereto as Exhibit M are true and correct copies of LANCIUM00033741 (May 10, 2019 email from McNamara), produced by Defendants in this case.

17. Attached hereto as Exhibit N is a true and correct copy of LANCIUM00026299, produced by Defendants in this case.

18. Attached hereto as Exhibit O are true and correct copies of LANCIUM00028482-LANCIUM00028484 (2019-08-14 Addendum for Fixed Price, Fixed Volume Electricity between Calpine and Lancium), produced by Defendants in this case.

19. Attached hereto as Exhibit P are true and correct copies of LANCIUM00033064-LANCIUM00033065, produced by Defendants in this case.

20. Attached hereto as Exhibit Q are true and correct copies of LANCIUM00028485-LANCIUM00028519 (2019-08-19 Presentation, Lancium – Powering the Future of Computing), produced by Defendants in this case.

21. Attached hereto as Exhibit R is a true and correct copy of LANCIUM00033139, produced by Defendants in this case.

22. Attached hereto as Exhibit S are true and correct copies of excerpts from Shams Siddiqi's Report on ERCOT Market Mechanisms.

23. Attached hereto as Exhibit T are true and correct copies of an email correspondence between counsel for Plaintiffs and counsel for Defendants, including attachments.

24. Attached hereto as Exhibit U are true and correct copies of excerpts from the Deposition Transcript of Stanley McClellan.

25. Attached hereto as Exhibit V are true and correct copies of excerpts from the Deposition Transcript of Victor Henrique.

26. Attached hereto as Exhibit W are true and correct copies of excerpts from the Deposition Transcript of Shams Siddiqi.

27. Attached hereto as Exhibit X are true and correct copies of excerpts from the September 8, 2021 Hearing Transcript.

28. Attached hereto as Exhibit Y are true and correct copies of LANCIUM00002892-LANCIUM00003134, produced by Defendants in this case.

29. Attached hereto as Exhibit Z are true and correct copies of LANCIUM00035821, produced by Defendants in this case.

30. Attached hereto as Exhibit AA are true and correct copies of LANCIUM00036373, produced by Defendants in this case.

31. Attached hereto as Exhibit BB are true and correct copies of BB_SC00000001-BB_SC000000067, produced by Plaintiffs in this case.

32. Attached hereto as Exhibit CC are true and correct copies of Defendants' Second Supplemental Response to Interrogatory No. 3.

33. Attached hereto as Exhibit DD are true and correct copies of Plaintiffs' Response to Interrogatory No. 17.

I declare under penalty of perjury that the foregoing is true and correct.

Date: July 19, 2022

By: Chelsea Murray
Chelsea Murray

CERTIFICATE OF SERVICE

I hereby certify that on the 19th day of July, 2022, the attached **DECLARATION OF CHELSEA MURRAY IN SUPPORT OF PLAINTIFFS' BRIEF IN OPPOSITION TO DEFENDANTS' FIRST MOTION FOR SUMMARY JUDGMENT** was served upon the below-named counsel of record at the address and in the manner indicated:

Chad S.C. Stover, Esquire — VIA ELECTRONIC MAIL
Barnes & Thornburg LLP
222 Delaware Avenue, Suite 1200
Wilmington, DE 19801

Mark C. Nelson, Esquire — VIA ELECTRONIC MAIL
Barnes & Thornburg LLP
2121 N. Pearl Street, Suite 700
Dallas, TX 75201

Adam M. Kaufmann, Esquire — VIA ELECTRONIC MAIL
Barnes & Thornburg LLP
One N. Wacker Drive, Suite 4400
Chicago, IL 60606-2833

/s/ Andrew C. Mayo

Andrew C. Mayo

EXHIBIT A

REDACTED IN ITS ENTIRETY

EXHIBIT B

REDACTED IN ITS ENTIRETY

EXHIBIT C

UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

BEARBOX LLC AND AUSTIN STORMS **Plaintiffs,** **v.** **LANCIUM LLC, MICHAEL T. MCNAMARA, AND RAYMOND E. CLINE, JR.,** **Defendants.**	**CASE NO. 1:21-cv-00534-MN-CJB**

EXPERT REPORT OF SOURCE CODE EXPERT NIKOLAUS BAER

CONTAINS CONFIDENTIAL BUSINESS INFORMATION &
CONFIDENTIAL SOURCE CODE – ATTORNEY'S EYES ONLY INFORMATION –
SUBJECT TO PROTECTIVE ORDER

something else, but these dates further call into question whether the produced source code files even represent the state of Mr. Storms and BearBox's software and knowledge as of May 3, 2019.

i. PDU UI Interface

39. The BearBox source code production included implementations for a user interface for monitoring and manually controlling the state of what appeared to be PDUs across a network, such as turning the relays of the PDUs on or off. The user interface related source code, however, was not connected to, did not receive information from, and was not called upon by the other two categories of source code discussed below. In addition, the BearBox source code production did not include any source code for the operation of the PDUs themselves.

40. This category of source code includes, for example, the source code file ***BearBox_PDU_v1.1.py***.[5] As also shown in Exhibit C, this source code file is representative of similar functionality, and identical or substantially similar lines of source code found in the source code files: ***cgminer_watchdog_basic.py***,[6] ***Kermit_PDU.py***,[7] ***PDU_Controller_3x48.py***,[8] and ***PDU_master.py***.[9]

41. The source code file ***BearBox_PDU_v1.1.py*** utilizes the third-party PyQt5 application framework to implement a graphic user interface GUI, as seen from the import statements at lines 3- 6.[10] Line 7 imports the ***ModbusTcpClient*** to implement network communications using the Modbus protocol, a network communication protocol commonly used with hardware devices.[11] The instruction at line 12 reads in a configuration file named ***BearBox_PDU_config.ini***. This configuration file was not produced, so it is unclear whether it actually exists or not. The class ***MainWindow*** at lines 23-47 of ***BearBox_PDU_v1.1.py*** sets up a window with the title "***Power Control***

[5] BB_SC00000068
[6] BB_SC00000024
[7] BB_SC00000069
[8] BB_SC00000070
[9] BB_SC00000071
[10] "What is PyQt?" Riverbank Computing Limited. Online: https://www.riverbankcomputing.com/software/pyqt/ [Accessed April 28, 2022].
[11] "PyModbus – A Python Modbus Stack" Sanjay. Online: https://pymodbus.readthedocs.io/en/3.0.0/source/library/modules.html [Accessed April 28, 2022].

iv. Overall Functionality

64. The BearBox source code includes several versions of these three main functionalities: PDU interface, public data retrieval, and simulating Bitcoin mining profits. These functionalities are reflected in the diagram labeled *BearBox Automatic Miner Management System Version 1.0*.[46] As the diagram demonstrates, the system that the BearBox source code was modeling was one where the system would operate with power generation assets and the software would continually determine whether it was more profitable to use the electricity from the generation assets to mine Bitcoin or to sell the electricity at the day-ahead or real-time LMP pricing.

[46] BB00000092

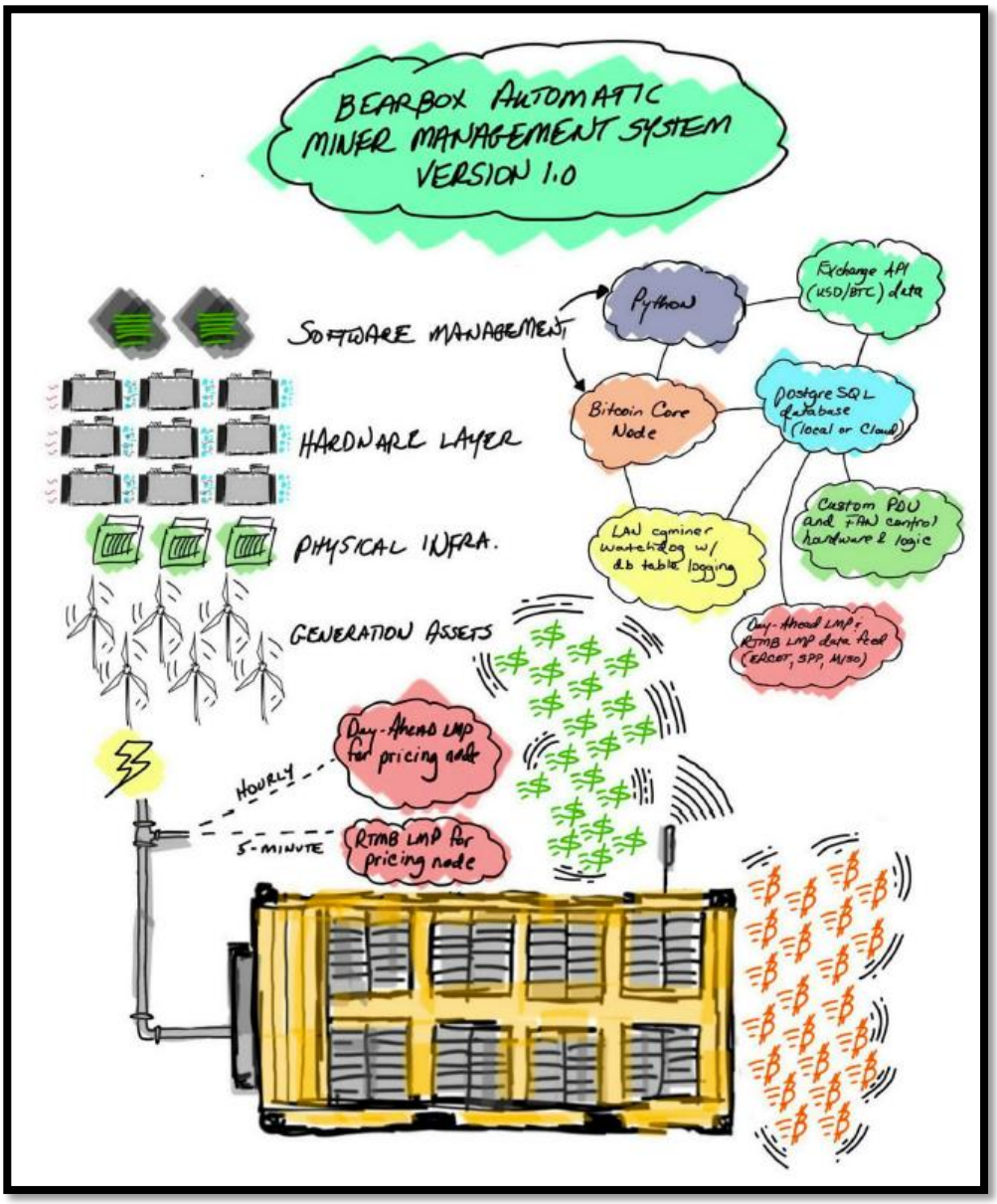


Figure 1. BearBox Automatic Miner Management System Version 1.0

65. My description and understanding of the source code for retrieving publicly available information to perform a simulation is consistent with Mr. Storms own description of his source code. For example, Mr. Storms also describes his work as just performing simulation:[47]

> Q. Now, at this point had you written anything that would connect you to the grid to allow you to sell any power back to it?

[47] 2/23/22 Austin Storms deposition at 164:5-165:23.

consumption) the BearBox source code simulating Bitcoin mining profits uses statically defined mock values, such as the variable ***miner_hashrate*** at line 29 of ***denis_logic.py.***[96]

111. In addition, although the file ***cgminer_sqlite_test.py*** requests and can receive data regarding the 5-second and 5-minute hash rate of a Bitcoin miner, this hash rate information is only used by ***cgminer_sqlite_test.py*** to determine whether the miner should be reset (i.e., if the 5-second hash rate is greater than the 5-minute hash rate) and this file and the hash rate data is not called upon or used by any other BearBox source code files. The hash rate information is not used to set or maintain a power consumption target, let alone a power consumption target that is equal to or greater than minimum power thresholds for specific time intervals that are specified by data received by the source code.

112. In paragraph 67 of the McClellan Report, Dr. McClellan identifies 11 BearBox source code files and makes the conclusory statement that these source code files "perform functions related to determining a performance strategy based on the power option data and monitored conditions."[97] Dr. McClellan does not explain what he means by this assertion, but for the reasons discussed above, I disagree that any of these files have the capability or functionality to "responsive to receiving the power option data, determine a performance strategy for the set of computing systems based on a combination of at least a portion of the power option data and at least one condition in the set of conditions, wherein the performance strategy comprises a power consumption target for the set of computing systems for each time interval in the set of time intervals, wherein each power consumption target is equal to or greater than the minimum power threshold associated with each time interval" as required by claim 1 of the '433 patent. None of the 11 identified source code files determine a performance strategy comprising a power consumption target for any time intervals where the power consumption target is equal to or greater than a minimum power threshold for a specific time interval and that is received by the system. Furthermore, no BearBox source code has the capability or functionality to determine or set a power consumption target for the system where the power consumption target is equal to or exceeds a minimum power threshold for a

[96] BB_SC00000004
[97] McClellan Report at ¶ 67.

source code does not support or demonstrate that Mr. Storms or BearBox conceived of or possessed the invention of claim 19.

xxii. Claim 20

145. Dr. McClellan does not provide any analysis regarding claim 20 and only refers back to his opinions and analysis regarding claim 1.[113] As such, for the all the reasons addressed above in my discussion of the BearBox source code in conjunction with claim 1, I disagree with Dr. McClellan, and it is my opinion that the BearBox source code does not support or demonstrate that Mr. Storms or BearBox conceived of or possessed the invention of claim 20.

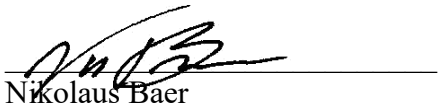
VIII. CONCLUSION

146. Based on my review of the BearBox source code, I disagree with Dr. McClellan, and it is my opinion that Mr. Storms and BearBox did not conceive of or possess the inventions claimed in the '433 patent and the BearBox source code does not support or indicate otherwise.

147. I declare under penalty of perjury that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true.

148. I understand that additional materials may be produced. I therefore reserve the right to supplement or amend my opinion, as expressed in this Report, following the production of additional materials and further analysis.

Executed on May 6, 2022, in Mountain View, CA.



Nikolaus Baer

[113] *See* McClellan Report at ¶¶ 166-169.

EXHIBIT G: CGMINER README FILE

CGMINER README (https://github.com/ckolivas/cgminer/blob/master/API-README)

This repository has been archived by the owner. It is now read-only.

ckolivas / cgminer Public archive

forked from jgarzik/cpuminer

Code Pull requests 14 Actions Projects Security Insights

master cgminer / API-README Go to file ...

kanoi API add Work Difficulty to pools Latest commit 2a47372 on Aug 13, 2015 History

6 contributors

1864 lines (1355 sloc) | 62.3 KB Raw Blame

```
This README contains details about the cgminer RPC API

It also includes some detailed information at the end,
about using miner.php


If you start cgminer with the "--api-listen" option, it will listen on a
simple TCP/IP socket for single string API requests from the same machine
running cgminer and reply with a string and then close the socket each time
If you add the "--api-network" option, it will accept API requests from any
network attached computer.

You can only access the comands that reply with data in this mode.
By default, you cannot access any privileged command that affects the miner -
you will receive an access denied status message see --api-allow below.

You can specify IP addresses/prefixes that are only allowed to access the API
with the "--api-allow" option e.g. --api-allow W:192.168.0.1,10.0.0/24
will allow 192.168.0.1 or any address matching 10.0.0.*, but nothing else
IP addresses are automatically padded with extra '.0's as needed
Without a /prefix is the same as specifying /32
0/0 means all IP addresses.
The 'W:' on the front gives that address/subnet privileged access to commands
that modify cgminer (thus all API commands)
Without it those commands return an access denied status.
See --api-groups below to define other groups like W:
Privileged access is checked in the order the IP addresses were supplied to
"--api-allow"
The first match determines the privilege level.
Using the "--api-allow" option overides the "--api-network" option if they
are both specified
With "--api-allow", 127.0.0.1 is not by default given access unless specified

If you start cgminer also with the "--api-mcast" option, it will listen for
a multicast message and reply to it with a message containing it's API port
number, but only if the IP address of the sender is allowed API access

More groups (like the privileged group W:) can be defined using the
--api-groups command
Valid groups are only the letters A-Z (except R & W are predefined) and are
not case sensitive
The R: group is the same as not privileged access
The W: group is (as stated) privileged access (thus all API commands)
To give an IP address/subnet access to a group you use the group letter
in front of the IP address instead of W: e.g. P:192.168.0/32
An IP address/subnet can only be a member of one group
A sample API group would be:
 --api-groups
        P:switchpool:enablepool:addpool:disablepool:removepool:poolpriority:*
This would create a group 'P' that can do all current pool commands and all
non-priviliged commands - the '*' means all non-priviledged commands
Without the '*' the group would only have access to the pool commands
Defining multiple groups example:
 --api-groups Q:quit:restart:*,S:save
This would define 2 groups:
 Q: that can 'quit' and 'restart' as well as all non-priviledged commands
 S: that can only 'save' and no other commands

The RPC API request can be either simple text or JSON.

If the request is JSON (starts with '{'), it will reply with a JSON formatted
response, otherwise it replies with text formatted as described further below.

The JSON request format required is '{"command":"CMD","parameter":"PARAM"}'
(though of course parameter is not required for all requests)
where "CMD" is from the "Request" column below and "PARAM" would be e.g.
the ASC/PGA number if required.

An example request in both formats to disable Hotplug:
```

```
  hotplug|0
  {"command":"hotplug","parameter":"0"}

The format of each reply (unless stated otherwise) is a STATUS section
followed by an optional detail section

From API version 1.7 onwards, reply strings in JSON and Text have the
necessary escaping as required to avoid ambiguity - they didn't before 1.7
For JSON the 2 characters '"' and '\' are escaped with a '\' before them
For Text the 4 characters '|' ',' '=' and '\' are escaped the same way

Only user entered information will contain characters that require being
escaped, such as Pool URL, User and Password or the Config save filename,
when they are returned in messages or as their values by the API

For API version 1.4 and later:

The STATUS section is:

 STATUS=X,When=NNN,Code=N,Msg=string,Description=string|

  STATUS=X Where X is one of:
   W - Warning
   I - Informational
   S - Success
   E - Error
   F - Fatal (code bug)

  When=NNN
   Standard long time of request in seconds

  Code=N
   Each unique reply has a unique Code (See api.c - #define MSG_NNNNNN)

  Msg=string
   Message matching the Code value N

  Description=string
   This defaults to the cgminer version but is the value of --api-description
   if it was specified at runtime.

With API V3.1 you can also request multiple report replies in a single command
request
e.g. to request both summary and devs, the command would be summary+devs

This is only available for report commands that don't need parameters,
and is not available for commands that change anything
Any parameters supplied will be ignored

The extra formatting of the result is to have a section for each command
e.g. CMD=summary|STATUS=....|CMD=devs|STATUS=...
With JSON, each result is within a section of the command name
e.g. {"summary":{"STATUS":[{"STATUS":"S"...}],"SUMMARY":[...],"id":1},
      "devs":{"STATUS":[{"STATUS:"S"...}],"DEVS":[...],"id":1},"id":1}

As before, if you supply bad JSON you'll just get a single 'E' STATUS section
in the old format, since it doesn't switch to using the new format until it
correctly processes the JSON and can match a '+' in the command

If you request a command multiple times, e.g. devs+devs
you'll just get it once
If this results in only one command, it will still use the new layout
with just the one command

If you request a command that can't be used due to requiring parameters,
a command that isn't a report, or an invalid command, you'll get an 'E' STATUS
for that one but it will still attempt to process all other commands supplied

Blank/missing commands are ignored e.g. +devs++
will just show 'devs' using the new layout

For API version 1.10 and later:

The list of requests - a (*) means it requires privileged access - and replies:

 Request       Reply Section  Details
 -------       -------------  -------
 version       VERSION        CGMiner=cgminer, version
                              API=API| version

 config        CONFIG         Some miner configuration information:
                              ASC Count=N, <- the number of ASCs
                              PGA Count=N, <- the number of PGAs
                              Pool Count=N, <- the number of Pools
                              Strategy=Name, <- the current pool strategy
                              Log Interval=N, <- log interval (--log N)
                              Device Code=ICA , <- spaced list of compiled
                                                   device drivers
                              OS=Linux/Apple/..., <- operating System

```

```
summary       SUMMARY        The status summary of the miner            \
                             e.g. Elapsed=NNN,Found Blocks=N,Getworks=N,...|

pools         POOLS          The status of each pool e.g.
                             Pool=0,URL=http://pool.com:6311,Status=Alive,...|

devs          DEVS           Each available PGA and ASC with their details
                             e.g. ASC=0,Accepted=NN,MHS av=NNN,...,Intensity=D|
                             Last Share Time=NNN, <- standand long time in sec
                              (or 0 if none) of last accepted share
                             Last Share Pool=N, <- pool number (or -1 if none)
                             Last Valid Work=NNN, <- standand long time in sec
                              of last work returned that wasn't an HW:
                             Will not report PGAs if PGA mining is disabled
                             Will not report ASCs if ASC mining is disabled

edevs[|old]   DEVS           The same as devs, except it ignores blacklisted
                             devices and zombie devices
                             If you specify the optional 'old' parameter, then
                             the output will include zombie devices that became
                             zombies less than 'old' seconds ago
                             A value of zero for 'old', which is the default,
                             means ignore all zombies
                             It will return an empty list of devices if all
                             devices are blacklisted or zombies

pga|N         PGA            The details of a single PGA number N in the same
                             format and details as for DEVS
                             This is only available if PGA mining is enabled
                             Use 'pgacount' or 'config' first to see if there
                             are any

pgacount      PGAS           Count=N| <- the number of PGAs
                             Always returns 0 if PGA mining is disabled

switchpool|N (*)
              none           There is no reply section just the STATUS section
                             stating the results of switching pool N to the
                             highest priority (the pool is also enabled)
                             The Msg includes the pool URL

enablepool|N (*)
              none           There is no reply section just the STATUS section
                             stating the results of enabling pool N
                             The Msg includes the pool URL

addpool|URL,USR,PASS (*)
              none           There is no reply section just the STATUS section
                             stating the results of attempting to add pool N
                             The Msg includes the pool number and URL
                             Use '\\' to get a '\' and '\,' to include a comma
                             inside URL, USR or PASS

poolpriority|N,... (*)
              none           There is no reply section just the STATUS section
                             stating the results of changing pool priorities
                             See usage below

poolquota|N,Q (*)
              none           There is no reply section just the STATUS section
                             stating the results of changing pool quota to Q

disablepool|N (*)
              none           There is no reply section just the STATUS section
                             stating the results of disabling pool N
                             The Msg includes the pool URL

removepool|N (*)
              none           There is no reply section just the STATUS section
                             stating the results of removing pool N
                             The Msg includes the pool URL
                             N.B. all details for the pool will be lost

save|filename (*)
              none           There is no reply section just the STATUS section
                             stating success or failure saving the cgminer
                             config to filename
                             The filename is optional and will use the cgminer
                             default if not specified

quit (*)      none           Status is a single "BYE" reply before cgminer
                             quits

notify        NOTIFY         The last status and history count of each devices
                             problem
                             This lists all devices including those not
                             supported by the 'devs' command e.g.
                             NOTIFY=0,Name=ASC,ID=0,Last Well=1332432290,...|

privileged (*)
```

```
                none            There is no reply section just the STATUS section
                                stating an error if you do not have privileged
                                access to the API and success if you do have
                                privilege
                                The command doesn't change anything in cgminer

pgaenable|N (*)
                none            There is no reply section just the STATUS section
                                stating the results of the enable request
                                You cannot enable a PGA if it's status is not WELL
                                This is only available if PGA mining is enabled

pgadisable|N (*)
                none            There is no reply section just the STATUS section
                                stating the results of the disable request
                                This is only available if PGA mining is enabled

pgaidentify|N (*)
                none            There is no reply section just the STATUS section
                                stating the results of the identify request
                                This is only available if PGA mining is enabled
                                and currently only BFL singles and Cairnsmore1's
                                with the appropriate firmware support this command
                                On a BFL single it will flash the led on the front
                                of the device for appoximately 4s
                                All other non BFL,ICA PGA devices will return a
                                warning status message stating that they dont
                                support it. Non-CMR ICAs will ignore the command.
                                This adds a 4s delay to the BFL share being
                                processed so you may get a message stating that
                                procssing took longer than 7000ms if the request
                                was sent towards the end of the timing of any work
                                being worked on
                                e.g.: BFL0: took 8438ms - longer than 7000ms
                                You should ignore this

devdetails      DEVDETAILS      Each device with a list of their static details
                                This lists all devices including those not
                                supported by the 'devs' command
                                e.g. DEVDETAILS=0,Name=ASC,ID=0,Driver=yuu,...|

restart (*)     none            Status is a single "RESTART" reply before cgminer
                                restarts

stats           STATS           Each device or pool that has 1 or more getworks
                                with a list of stats regarding getwork times
                                The values returned by stats may change in future
                                versions thus would not normally be displayed
                                Device drivers are also able to add stats to the
                                end of the details returned

estats[|old]    STATS           The same as stats, except it ignores blacklisted
                                devices, zombie devices and pools
                                If you specify the optional 'old' parameter, then
                                the output will include zombie devices that became
                                zombies less than 'old' seconds ago
                                A value of zero for 'old', which is the default,
                                means ignore all zombies
                                It will return an empty list of devices if all
                                devices are blacklisted or zombies

check|cmd       CHECK           Exists=Y/N, <- 'cmd' exists in this version
                                Access=Y/N| <- you have access to use 'cmd'

failover-only|true/false (*)
                none            This command has been deprecated, only returning a
                                deprecated message.

coin            COIN            Coin mining information:
                                Hash Method=sha256/scrypt,
                                Current Block Time=N.N, <- 0 means none
                                Current Block Hash=XXXX..., <- blank if none
                                LP=true/false, <- LP is in use on at least 1 pool
                                Network Difficulty=NN.NN|

debug|setting (*)
                DEBUG           Debug settings
                                The optional commands for 'setting' are the same
                                as the screen curses debug settings
                                You can only specify one setting
                                Only the first character is checked - case
                                insensitive:
                                Silent, Quiet, Verbose, Debug, RPCProto,
                                PerDevice, WorkTime, Normal
                                The output fields are (as above):
                                Silent=true/false,
                                Quiet=true/false,
                                Verbose=true/false,
                                Debug=true/false,
                                RPCProto=true/false,
```

```
                              PerDevice=true/false,                    \
                              WorkTime=true/false|

setconfig|name,N (*)
                 none         There is no reply section just the STATUS section
                              stating the results of setting 'name' to N
                              No values are supported any more and only a
                              deprecated message will be returned.

usbstats      USBSTATS       Stats of all LIBUSB mining devices except ztex
                              e.g. Name=MMQ,ID=0,Stat=SendWork,Count=99,...|

pgaset|N,opt[,val] (*)
                 none         There is no reply section just the STATUS section
                              stating the results of setting PGA N with
                              opt[,val]
                              This is only available if PGA mining is enabled

                              If the PGA does not support any set options, it
                              will always return a WARN stating pgaset isn't
                              supported

                              If opt=help it will return an INFO status with a
                              help message about the options available

                              The current options are:
                               MMQ opt=clock val=160 to 230 (a multiple of 2)
                               CMR opt=clock val=100 to 220

zero|Which,true/false (*)
                 none         There is no reply section just the STATUS section
                              stating that the zero, and optional summary, was
                              done
                              If Which='all', all normal cgminer and API
                              statistics will be zeroed other than the numbers
                              displayed by the usbstats and stats commands
                              If Which='bestshare', only the 'Best Share' values
                              are zeroed for each pool and the global
                              'Best Share'
                              The true/false option determines if a full summary
                              is shown on the cgminer display like is normally
                              displayed on exit.

hotplug|N (*) none            There is no reply section just the STATUS section
                              stating that the hotplug setting succeeded
                              If the code is not compiled with hotplug in it,
                              the the warning reply will be
                               'Hotplug is not available'
                              If N=0 then hotplug will be disabled
                              If N>0 && <=9999, then hotplug will check for new
                              devices every N seconds

asc|N         ASC            The details of a single ASC number N in the same
                              format and details as for DEVS
                              This is only available if ASC mining is enabled
                              Use 'asccount' or 'config' first to see if there
                              are any

ascenable|N (*)
                 none         There is no reply section just the STATUS section
                              stating the results of the enable request
                              You cannot enable a ASC if it's status is not WELL
                              This is only available if ASC mining is enabled

ascdisable|N (*)
                 none         There is no reply section just the STATUS section
                              stating the results of the disable request
                              This is only available if ASC mining is enabled

ascidentify|N (*)
                 none         There is no reply section just the STATUS section
                              stating the results of the identify request
                              This is only available if ASC mining is enabled
                              and currently only BFL ASICs support this command
                              On a BFL single it will flash the led on the front
                              of the device for appoximately 4s
                              All other non BFL ASIC devices will return a
                              warning status message stating that they dont
                              support it

asccount      ASCS           Count=N| <- the number of ASCs
                              Always returns 0 if ASC mining is disabled

ascset|N,opt[,val] (*)
                 none         There is no reply section just the STATUS section
                              stating the results of setting ASC N with
                              opt[,val]
                              This is only available if ASC mining is enabled

                              If the ASC does not support any set options, it
```

```
                              will always return a WARN stating ascset isn't
                              supported

                              If opt=help it will return an INFO status with a
                              help message about the options available

                              The current options are:
                               AVA+BTB opt=freq val=256 to 1024 - chip frequency
                               BTB opt=millivolts val=1000 to 1400 - corevoltage
                               MBA opt=reset val=0 to chipcount - reset a chip
                               BMA opt=volt val=0-9 opt=clock val=0-15
                               MBA opt=freq val=0-chip:100-1400 - set chip freq
                               MBA opt=ledcount val=0-100 - chip count for led
                               MBA opt=ledlimit val=0-200 - led off below GHs
                               MBA opt=spidelay val=0-9999 - SPI per I/O delay
                               MBA opt=spireset i|s0-9999 - SPI regular reset
                               MBA opt=spisleep val=0-9999 - SPI reset sleep ms

 lcd           LCD             An all-in-one short status summary of the miner
                               e.g. Elapsed,GHS av,GHS 5m,GHS 5s,Temp,
                                    Last Share Difficulty,Last Share Time,
                                    Best Share,Last Valid Work,Found Blocks,
                                    Pool,User|

 lockstats (*) none            There is no reply section just the STATUS section
                               stating the results of the request
                               A warning reply means lock stats are not compiled
                               into cgminer
                               The API writes all the lock stats to stderr

When you enable, disable or restart a PGA or ASC, you will also get
Thread messages in the cgminer status window

The 'poolpriority' command can be used to reset the priority order of multiple
pools with a single command - 'switchpool' only sets a single pool to first
priority
Each pool should be listed by id number in order of preference (first = most
preferred)
Any pools not listed will be prioritised after the ones that are listed, in the
priority order they were originally
If the priority change affects the miner's preference for mining, it may switch
immediately

When you switch to a different pool to the current one (including by priority
change), you will get a 'Switching to URL' message in the cgminer status
windows

Obviously, the JSON format is simply just the names as given before the '='
with the values after the '='

If you enable cgminer debug (-D or --debug) or, when cgminer debug is off,
turn on debug with the API command 'debug|debug' you will also get messages
showing some details of the requests received and the replies

There are included 4 program examples for accessing the API:

api-example.php - a php script to access the API
  usAge: php api-example.php command
 by default it sends a 'summary' request to the miner at 127.0.0.1:4028
 If you specify a command it will send that request instead
 You must modify the line "$socket = getsock('127.0.0.1', 4028);" at the
 beginning of "function request($cmd)" to change where it looks for cgminer

api-example.rb - a Ruby script to access the API.
 usage: ruby api-example.rb command[:parameter] [HOST [PORT]]
This script prints the parsed cgminer API response

API.java/API.class
 a java program to access the API (with source code)
  usAge is: java API command address port
 Any missing or blank parameters are replaced as if you entered:
  java API summary 127.0.0.1 4028

api-example.c - a 'C' program to access the API (with source code)
  usAge: api-example [command [ip/host [port]]]
 again, as above, missing or blank parameters are replaced as if you entered:
  api-example summary 127.0.0.1 4028

miner.php - an example web page to access the API
 This includes buttons and inputs to attempt access to the privileged commands
 See the end of this API-README for details of how to tune the display
 and also to use the option to display a multi-rig summary

----------

Feature Changelog for external applications using the API:

---------

API V3.7 (cgminer v4.9.3?)
```

```
                                                                              \

Modified API commands:
 'pools' - add 'Work Difficulty'

----------

API V3.6 (cgminer v4.9.2)

Modified API commands:
 'pools' - add 'Bad Work'
 'setconfig' - expire, scantime and queue have all been deprecated and will
 only return a deprecated message.
 'failover-only' - deprecated, a deprecated message will be returned.

----------

API V3.5 (cgminer v4.7.0)

- Made quit and restart return valid JSON as a STATUS mirroring the request.
- Made addpool return what pool number the added pool is.

----------

API V3.4 (cgminer v4.3.?)

Added API commands:
 'lcd' - An all-in-one short status summary of the miner

----------

API V3.3 (cgminer v4.2.0)

Added API commands:
 'edevs' - Only enabled devices, for 'devs'
 'estats' - Only enabled devices, for 'stats'

----------

API V3.2 (cgminer v4.1.0)

Fix for:
HEX32 data type in the API version v3.1 JSON - since cgminer v3.12.1 -
returns an incorrect formatted json data element for the API stats command
for HashFast hardware

----------

API V3.1 (cgminer v3.12.1)

Multiple report request command with '+' e.g. summary+devs

----------

API V3.0 (cgminer v3.11.0)

Allow unlimited size replies

----------

API V2.0 (cgminer v3.8.0)

Removed all GPU related commands and information from the replies

----------

API V1.32 (cgminer v3.6.5)

Modified API commands:
 'devs' 'gpu' 'pga' and 'asc' - add 'Device Elapsed'

----------

API V1.31 (cgminer v3.6.3)

Added API command:
 'lockstats' - display cgminer dev lock stats if compiled in

Modified API command:
 'summary' - add 'MHS %ds' (where %d is the log interval)

----------

API V1.30 (cgminer v3.4.3)

Added API command:
 'poolquota' - Set pool quota for load-balance strategy.

Modified API command:
 'pools' - add 'Quota'

```

```
---------

API V1.29 (cgminer v3.4.1)

Muticast identification added to the API

----------

API V1.28 (cgminer v3.3.4)

Modified API commands:
 'devs', 'pga', 'asc', 'gpu' - add 'Device Hardware%' and 'Device Rejected%'
 'pools' - add 'Pool Rejected%' and 'Pool Stale%'
 'summary' - add 'Device Hardware%', 'Device Rejected%', 'Pool Rejected%',
                 'Pool Stale%'

----------

API V1.27 (cgminer v3.3.2)

Added API commands:
 'ascset' - with: BTB opt=millivolts val=1000 to 1310 - core voltage
                  AVA+BTB opt=freq val=256 to 450 - chip frequency

----------

API V1.26 (cgminer v3.2.3)

Remove all CPU support (cgminer v3.0.0)

Added API commands:
 'asc'
 'ascenable'
 'ascdisable'
 'ascidentify|N' (only works for BFL ASICs so far)
 'asccount'

Various additions to the debug 'stats' command

----------

API V1.25

Added API commands:
 'hotplug'

Modified API commands:
 'devs' 'gpu' and 'pga' - add 'Last Valid Work'
 'devs' - list ASIC devices
 'config' - add 'Hotplug', 'ASC Count'
 'coin' - add 'Network Difficulty'

----------

API V1.24 (cgminer v2.11.0)

Added API commands:
 'zero'

Modified API commands:
 'pools' - add 'Best Share'
 'devs' and 'pga' - add 'No Device' for PGAs if MMQ or BFL compiled
 'stats' - add pool: 'Net Bytes Sent', 'Net Bytes Recv'

----------

API V1.23 (cgminer v2.10.2)

Added API commands:
 'pgaset' - with: MMQ opt=clock val=160 to 230 (and a multiple of 2)

----------

API V1.22 (cgminer v2.10.1)

Enforced output limitation:
 all extra records beyond the output limit of the API (~64k) are ignored
  and chopped off at the record boundary before the limit is reached
  however, JSON brackets will be correctly closed and the JSON id will be
  set to 0 (instead of 1) if any data was truncated

Modified API commands:
 'stats' - add 'Times Sent', 'Bytes Sent', 'Times Recv', 'Bytes Recv'

----------

API V1.21 (cgminer v2.10.0)

Added API commands:
 'usbstats'
```

```
                                                                              \

Modified API commands:
 'summary' - add 'Best Share'

Modified output:
 each MMQ shows up as 4 devices, each with it's own stats

----------

API V1.20 (cgminer v2.8.5)

Modified API commands:
 'pools' - add 'Has Stratum', 'Stratum Active', 'Stratum URL'

----------

API V1.19 (cgminer v2.7.6)

Added API commands:
 'debug'
 'pgaidentify|N' (only works for BFL Singles so far)
 'setconfig|name,N'

Modified API commands:
 'devs' - add 'Diff1 Work', 'Difficulty Accepted', 'Difficulty Rejected',
             'Last Share Difficulty' to all devices
 'gpu|N' - add 'Diff1 Work', 'Difficulty Accepted',
              'Difficulty Rejected', 'Last Share Difficulty'
 'pga|N' - add 'Diff1 Work', 'Difficulty Accepted',
              'Difficulty Rejected', 'Last Share Difficulty'
 'notify' - add '*Dev Throttle' (for BFL Singles)
 'pools' - add 'Proxy Type', 'Proxy', 'Difficulty Accepted',
               'Difficulty Rejected', 'Difficulty Stale',
               'Last Share Difficulty'
 'config' - add 'Queue', 'Expiry'
 'stats' - add 'Work Diff', 'Min Diff', 'Max Diff', 'Min Diff Count',
               'Max Diff Count' to the pool stats

----------

API V1.18 (cgminer v2.7.4)

Modified API commands:
 'stats' - add 'Work Had Roll Time', 'Work Can Roll', 'Work Had Expire',
                'Work Roll Time' to the pool stats
 'config' - include 'ScanTime'

----------

API V1.17 (cgminer v2.7.1)

Added API commands:
 'coin'

Modified API commands:
 'summary' - add 'Work Utility'
 'pools' - add 'Diff1 Shares'

----------

API V1.16 (cgminer v2.6.5)

Added API commands:
 'failover-only'

Modified API commands:
 'config' - include failover-only state

----------

API V1.15 (cgminer v2.6.1)

Added API commands:
 'poolpriority'

----------

API V1.14 (cgminer v2.5.0)

Modified API commands:
 'stats' - more icarus timing stats added
 'notify' - include new device comms error counter

The internal code for handling data was rewritten (~25% of the code)
Completely backward compatible

----------

API V1.13 (cgminer v2.4.4)

```

```
Added API commands:
 'check'

Support was added to cgminer for API access groups with the --api-groups option
It's 100% backward compatible with previous --api-access commands

----------

API V1.12 (cgminer v2.4.3)

Modified API commands:
 'stats' - more pool stats added

Support for the ModMinerQuad FPGA was added

----------

API V1.11 (cgminer v2.4.2)

Modified API commands:
 'save' no longer requires a filename (use default if not specified)

'save' incorrectly returned status E (error) on success before.
It now correctly returns S (success)

----------

API V1.10 (cgminer v2.4.1)

Added API commands:
 'stats'

N.B. the 'stats' command can change at any time so any specific content
present should not be relied upon.
The data content is mainly used for debugging purposes or hidden options
in cgminer and can change as development work requires

Modified API commands:
 'pools' added "Last Share Time"

----------

API V1.9 (cgminer v2.4.0)

Added API commands:
 'restart'

Modified API commands:
 'notify' corrected invalid JSON

----------

API V1.8 (cgminer v2.3.5)

Added API commands:
 'devdetails'

Support for the ZTex FPGA was added

----------

API V1.7 (cgminer v2.3.4)

Added API commands:
 'removepool'

Modified API commands:
 'pools' added "User"

From API version 1.7 onwards, reply strings in JSON and Text have the
necessary escaping as required to avoid ambiguity
For JSON the 2 characters '"' and '\' are escaped with a '\' before them
For Text the 4 characters '|' ',' '=' and '\' are escaped the same way

----------

API V1.6 (cgminer v2.3.2)

Added API commands:
 'pga'
 'pgaenable'
 'pgadisable'
 'pgacount'

Modified API commands:
 'devs' now includes Icarus and Bitforce FPGA devices
 'notify' added "*" to the front of the name of all numeric error fields
 'config' correct "Log Interval" to use numeric (not text) type for JSON

Support for Icarus and Bitforce FPGAs was added
```

```
\

----------

API V1.5 was not released

----------

API V1.4 (Kano's interim release of cgminer v2.3.1)

Added API commands:
 'notify'

Modified API commands:
 'config' added "Device Code" and "OS"

Added "When" to the STATUS reply section of all commands

----------

API V1.3 (cgminer v2.3.1-2)

Added API commands:
 'addpool'

Modified API commands:
 'devs'/'gpu' added "Total MH" for each device
 'summary' added "Total MH"

----------

API V1.2 (cgminer v2.3.0)

Added API commands:
 'enablepool'
 'disablepool'
 'privileged'

Modified API commands:
 'config' added "Log Interval"

Starting with API V1.2, any attempt to access a command that requires
privileged security, from an IP address that does not have privileged
security, will return an "Access denied" Error Status

----------

API V1.1 (cgminer v2.2.4)

There were no changes to the API commands in cgminer v2.2.4,
however support was added to cgminer for IP address restrictions
with the --api-allow option

----------

API V1.1 (cgminer v2.2.2)

Prior to V1.1, devs/gpu incorrectly reported GPU0 Intensity for all GPUs

Modified API commands:
 'devs'/'gpu' added "Last Share Pool" and "Last Share Time" for each device

----------

API V1.0 (cgminer v2.2.0)

Remove default CPU support

Added API commands:
 'config'
 'gpucount'
 'cpucount'
 'switchpool'
 'gpuintensity'
 'gpumem'
 'gpuengine'
 'gpufan'
 'gpuvddc'
 'save'

----------

API V0.7 (cgminer v2.1.0)

Initial release of the API in the main cgminer git

Commands:
 'version'
 'devs'
 'pools'
 'summary'
```

```
 'gpuenable'
 'gpudisable'
 'gpurestart'
 'gpu'
 'cpu'
 'gpucount'
 'cpucount'
 'quit'

----------------------------------------

miner.php
=========

miner.php is a PHP based interface to the cgminer RPC API
(referred to simply as the API below)

It can show rig details, summaries and input fields to allow you to change
cgminer
You can also create custom summary pages with it

It has two levels to the security:
1) cgminer can be configured to allow or disallow API access and access level
   security for miner.php
2) miner.php can be configured to allow or disallow privileged cgminer
   access, if cgminer is configured to allow privileged access for miner.php

---------

To use miner.php requires a web server with PHP

Basics: On xubuntu 11.04, to install apache2 and php, the commands are:
 sudo apt-get install apache2
 sudo apt-get install php5
 sudo /etc/init.d/apache2 reload

On Fedora 17:
 yum install httpd php
 systemctl restart httpd.service
 systemctl enable httpd.service --system

On windows there are a few options.
Try one of these (apparently the first one is easiest - thanks jborkl)
 http://www.easyphp.org/
 http://www.apachefriends.org/en/xampp.html
 http://www.wampserver.com/en/

---------

The basic cgminer option to enable the API is:

 --api-listen

or in your cgminer.conf

 "api-listen" : true,

(without the ',' on the end if it is the last item)

If the web server is running on the cgminer computer, the above
is the only change required to give miner.php basic access to
the cgminer API

-

If the web server runs on a different computer to cgminer,
you will also need to tell cgminer to allow the web server
to access cgminer's API and tell miner.php where cgminer is

Assuming a.b.c.d is the IP address of the web server, you
would add the following to cgminer:

 --api-listen --api-allow a.b.c.d

or in your cgminer.conf

 "api-listen" : true,
 "api-allow" : "a.b.c.d",

to tell cgminer to give the web server read access to the API

You also need to tell miner.php where cgminer is.
Assuming cgminer is at IP address e.f.g.h, then you would
edit miner.php and change the line

 $rigs = array('127.0.0.1:4028');

to

 $rigs = array('e.f.g.h:4028');
```

```
                                                                        \

See --api-network or --api-allow for more access details
and how to give write access

You can however, also tell miner.php to find your cgminer rigs automatically
on the local subnet

Add the following to each cgminer:

 --api-mcast

or in your cgminer.conf

 "api-mcast" : true,

And in miner.php set $mcast = true;

This will ignore the value of $rigs and overwrite it with the list of zero or
more rigs found on the network in the timeout specified
A rig will not reply if the API settings would mean it would also ignore an
API request from the web server running miner.php

---------

Once you have a web server with PHP running

 copy your miner.php to the main web folder

On Xubuntu 11.04
 /var/www/

On Fedora 17
 /var/www/html/

On Windows
 see your windows Web/PHP documentation

Assuming the IP address of the web server is a.b.c.d
Then in your web browser go to:

 http://a.b.c.d/miner.php

Done :)

---------

The rest of this documentation deals with the more complex
functions of miner.php, using myminer.php, creaing custom
summaries and displaying multiple cgminer rigs

---------

If you create a file called myminer.php in the same web folder
where you put miner.php, miner.php will load it when it runs

This is useful, to put any changes you need to make to miner.php
instead of changing miner.php
Thus if you update/get a new miner.php, you won't lose the changes
you have made if you put all your changes in myminer.php
(and don't change miner.php at all)

A simple example myminer.php that defines 2 rigs
(that I will keep referring to further below) is:

<?php
#
$rigs = array('192.168.0.100:4028:A', '192.168.0.102:4028:B');
#
?>

Changes in myminer.php superscede what is in miner.php
However, this is only valid for variables in miner.php before the
2 lines where myminer.php is included by miner.php:

 if (file_exists('myminer.php'))
  include_once('myminer.php');

Every variable in miner.php above those 2 lines, can be changed by
simply defining them in your myminer.php

So although miner.php originally contains the line

 $rigs = array('127.0.0.1:4028');

if you created the example myminer.php given above, it would actually
change the value of $rigs that is used when miner.php is running
i.e. you don't have to remove or comment out the $rigs line in miner.php
It will be superceded by myminer.php

---------
```

```
                                                                    \
The example myminer.php above also shows how to define more that one rig
to be shown my miner.php

Each rig string is 2 or 3 values seperated by colons ':'
They are simply an IP address or host name, followed by the
port number (usually 4028) and an optional Name string

miner.php displays rig buttons that will show the defails of a single
rig when you click on it - the button shows either the rig number,
or the 'Name' string if you provide it

PHP arrays contain each string seperated by a comma, but no comma after
the last one

So an example for 3 rigs would be:

 $rigs = array('192.168.0.100:4028:A', '192.168.0.102:4028:B',
               '192.168.0.110:4028:C');

Of course each of the rigs listed would also have to have the API
running and be set to allow the web server to access the API - as
explained before

---------

So basically, any variable explained below can be put in myminer.php
if you wanted to set it to something different to it's default value
and did not want to change miner.php itself every time you updated it

Below is each variable that can be changed and an explanation of each

---------

Default:
 $dfmt = 'H:i:s j-M-Y \U\T\CP';

Define the date format used to print full length dates
If you get the string 'UTCP' on the end of your dates shown, that
means you are using an older version of PHP and you can instead use:
 $dfmt = 'H:i:s j-M-Y \U\T\CO';

The PHP documentation on the date format is here:
 http://us.php.net/manual/en/function.date.php

---------

Default:
 $title = 'Mine';

Web page title
If you know PHP you can of course use code to define it e.g.
 $title = 'My Rig at: '.date($dfmt);

Which would set the web page title to something like:
 My Rig at: 10:34:00 22-Aug-2012 UTC+10:00

---------

Default:
 $readonly = false;

Set $readonly to true to force miner.php to be readonly
This means it won't allow you to change cgminer even if the cgminer API
options allow it to

If you set $readonly to false then it will check cgminer 'privileged'
and will show input fields and buttons on the single rig page
allowing you to change devices, pools and even quit or restart
cgminer

However, if the 'privileged' test fails, the code will set $readonly to
true

---------

Default:
 $userlist = null;

Define password checking and default access
 null means there is no password checking

$userlist is an array of 3 arrays e.g.
$userlist = array('sys' => array('boss' => 'bpass'),
                  'usr' => array('user' => 'upass', 'pleb' => 'ppass'),
                  'def' => array('Pools'));

'sys' is an array of system users and passwords (full access)
'usr' is an array of user level users and passwords (readonly access)
'def' is an array of custompages that anyone not logged in can view
```

```
\

Any of the 3 can be null, meaning there are none of that item

All validated 'usr' users are given $readonly = true; access
All validated 'sys' users are given the $readonly access you defined

If 'def' has one or more values, and allowcustompages is true, then
anyone without a password can see the list of custompage buttons given
in 'def' and will see the first one when they go to the web page, with
a login button at the top right

From the login page, if you login with no username or password, it will
show the first 'def' custompage (if there are any)

If you are logged in, it will show a logout button at the top right

---------

Default:
 $notify = true;

Set $notify to false to NOT attempt to display the notify command
table of data

Set $notify to true to attempt to display the notify command on
the single rig page
If your older version of cgminer returns an 'Invalid command'
coz it doesn't have notify - it just shows the error status table

---------

Default:
 $checklastshare = true;

Set $checklastshare to true to do the following checks:
If a device's last share is 12x expected ago then display as an error
If a device's last share is 8x expected ago then display as a warning
If either of the above is true, also display the whole line highlighted
This assumes shares are 1 difficulty shares

Set $checklastshare to false to not do the above checks

'expected' is calculated from the device MH/s value
So for example, a device that hashes at 380MH/s should (on average)
find a share every 11.3s
If the last share was found more than 11.3 x 12 seconds (135.6s) ago,
it is considered an error and highlighted
If the last share was found more than 11.3 x 8 seconds (90.4s) ago,
it is considered a warning and highlighted

The default highlighting is very subtle

---------

Default:
 $poolinputs = false;

Set $poolinputs to true to show the input fields for adding a pool
and changing the pool priorities on a single rig page
However, if $readonly is true, it will not display them

---------

Default:
 $rigport = 4028;

Default port to use if any $rigs entries don't specify the port number

---------

Default:
 $rigs = array('127.0.0.1:4028');

Set $rigs to an array of your cgminer rigs that are running
 format: 'IP' or 'Host' or 'IP:Port' or 'Host:Port' or 'Host:Port:Name'
If you only have one rig, it will just show the detail of that rig
If you have more than one rig it will show a summary of all the rigs
 with buttons to show the details of each rig -
 the button contents will be 'Name' rather than rig number, if you
 specify 'Name'
If Port is missing or blank, it will try $rigport
e.g. $rigs = array('127.0.0.1:4028','myrig.com:4028:Sugoi');

---------

Default:
 $rignames = false;

Set $rignames to false to not affect the display.
Set $rignames to one of 'ip' or 'ipx' to alter the name displayed
```

```
if the rig doesn't have a 'name' in $rigs
Currently:
 'ip' means use the 4th byte of the rig IP address as an integer
 'ipx' means use the 4th byte of the rig IP address as 2 hex bytes

---------

Default:
 $rigbuttons = true;

Set $rigbuttons to false to display a link rather than a button on
the left of any summary table with rig buttons, in order to reduce
the height of the table cells

---------

Default:
 $mcast = false;

Set $mcast to true to look for your rigs and ignore $rigs

---------

Default:
 $mcastexpect = 0;

The minimum number of rigs expected to be found when $mcast is true
If fewer are found, an error will be included at the top of the page

---------

Default:
 $mcastaddr = '224.0.0.75';

API Multicast address all cgminers are listening on

---------

Default:
 $mcastport = 4028;

API Multicast UDP port all cgminers are listening on

---------

Default:
 $mcastcode = 'FTW';

The code all cgminers expect in the Multicast message sent
The message sent is "cgm-code-listport"
Don't use the '-' character if you change it

---------

Default:
 $mcastlistport = 4027;

UDP port number that is added to the broadcast message sent
that specifies to the cgminers the port to reply on

---------

Default:
 $mcasttimeout = 1.5;

Set $mcasttimeout to the number of seconds (floating point)
to wait for replies to the Multicast message
N.B. the accuracy of the timing used to wait for the replies is
~0.1s so there's no point making it more than one decimal place

---------

Default:
 $mcastretries = 0;

Set $mcastretries to the number of times to retry the multicast

If $mcastexpect is 0, this is simply the number of extra times
that it will send the multicast request
N.B. cgminer doesn't listen for multicast requests for 1000ms after
each one it hears

If $mcastexpect is > 0, it will stop looking for replies once it
has found at least $mcastexpect rigs, but it only checks this rig
limit each time it reaches the $mcasttimeout limit, thus it can find
more than $mcastexpect rigs if more exist
It will send the multicast message up to $mcastretries extra times or
until it has found at least $mcastexpect rigs
However, when using $mcastretries, it is possible for it to sometimes
ignore some rigs on the network if $mcastexpect is less than the
```

```
number of rigs on the network and some rigs are too slow to reply          \

---------

Default:
 $allowgen = false;

Set $allowgen to true to allow customsummarypages to use 'gen' and 'bgen'
false means ignore any 'gen' or 'bgen' options
This is disabled by default due to the possible security risk of using it
See the end of this document for an explanation

---------

Default:
 $rigipsecurity = true;

Set $rigipsecurity to false to show the IP/Port of the rig
in the socket error messages and also show the full socket message

---------

Default:
 $rigtotals = true;
 $forcerigtotals = false;

Set $rigtotals to true to display totals on the single rig page
'false' means no totals (and ignores $forcerigtotals)

If $rigtotals is true, all data is also right aligned
With false, it's as before, left aligned

This option is just here to allow people to set it to false
if they prefer the old non-total display when viewing a single rig

Also, if there is only one line shown in any section, then no
total will be shown (to save screen space)
You can force it to always show rig totals on the single rig page,
even if there is only one line, by setting $forcerigtotals = true;

---------

Default:
 $socksndtimeoutsec = 10;
 $sockrcvtimeoutsec = 40;

The numbers are integer seconds

The defaults should be OK for most cases
However, the longer SND is, the longer you have to wait while
php hangs if the target cgminer isn't runnning or listening

RCV should only ever be relevant if cgminer has hung but the
API thread is still running, RCV would normally be >= SND

Feel free to increase SND if your network is very slow
or decrease RCV if that happens often to you

Also, on some windows PHP, apparently the $usec is ignored
(so usec can't be specified)

---------

Default:
 $hidefields = array();

List of fields NOT to be displayed
You can use this to hide data you don't want to see or don't want
shown on a public web page
The list of sections are:
 SUMMARY, POOL, PGA, GPU, NOTIFY, CONFIG, DEVDETAILS, DEVS
See the web page for the list of field names (the table headers)
It is an array of 'SECTION.Field Name' => 1

This example would hide the slightly more sensitive pool information:
Pool URL and pool username:
 $hidefields = array('POOL.URL' => 1, 'POOL.User' => 1);

If you just want to hide the pool username:
 $hidefields = array('POOL.User' => 1);

---------

Default:
 $ignorerefresh = false;
 $changerefresh = true;
 $autorefresh = 0;

Auto-refresh of the page (in seconds) - integers only
```

```
$ignorerefresh = true/false always ignore refresh parameters
$changerefresh = true/false show buttons to change the value
$autorefresh = default value, 0 means dont auto-refresh

---------

Default:
 $miner_font_family = 'verdana,arial,sans';
 $miner_font_size = '13pt';

Change these to set the font and font size used on the web page

---------

Default:
 $add_css_names = array();

List of CSS names to add to the CSS style object
        e.g. array('td.cool' => false);
true/false to not include the default $miner_font
The CSS name/value pairs must be defined in $colouroverride below

This allows you to create multiple complete CSS styles, optionally
using a different font to the default used/specified for all other
styles, and then when using the class name in a custom formatting
function (fmt) in a customsummarypage, it can use this style

---------

Default:
 $colouroverride = array();

Use this to change the web page colour scheme

See $colourtable in miner.php for the list of possible names to change

Simply put in $colouroverride, just the colours you wish to change

e.g. to change the colour of the header font and background
you could do the following:

 $colouroverride = array(
        'td.h color'            => 'green',
        'td.h background'       => 'blue'
 );

You can also add your own CSS styles to be used by a customsummarypage
custom format function, if you specify the class name in $add_css_names
and put the class styles in $colouroverride

---------

Default:
 $placebuttons = 'top';

Where to place the Refresh, Summary, Custom Pages, Quit, etc. buttons

Valid values are: 'top' 'bot' 'both'
 anything else means don't show them - case sensitive

---------

Default:
 $allowcustompages = true;

Should we allow custom pages?
(or just completely ignore them and don't display the buttons)

---------

OK this part is more complex: Custom Summary Pages

A custom summary page in an array of 'section' => array('FieldA','FieldB'...)

The section defines what data you want in the summary table and the Fields
define what data you want shown from that section

Standard sections are:
 SUMMARY, POOL, PGA, GPU, NOTIFY, CONFIG, DEVDETAILS, DEVS, EDEVS, STATS,
 ESTATS, COIN

Fields are the names as shown on the headers on the normal pages

There is a special field name '#' that will total to the number of rows
displayed in the custom summary page
In the actual row output it is a row counter per rig

Fields can be 'name=new name' to display 'name' with a different heading
'new name'
```

```
There are also now joined sections:                                    \
 SUMMARY+POOL, SUMMARY+DEVS, SUMMARY+EDEVS, DEVS+STATS, EDEVS+ESTATS,
 POOL+STATS plus many more
See the miner.php function joinsections() for the full list

These sections are an SQL join of the two sections and the fields in them
are named section.field where section. is the section the field comes from
See the example further down

Also note:
- empty tables are not shown
- empty columns (e.g. an unknown field) are not shown
- missing field data shows as blank
- the field name '*' matches all fields except in joined sections
  (useful for STATS and COIN)

There are 2 hard coded sections:
 DATE - displays a date table like at the start of 'Summary'
 RIGS - displays a rig table like at the start of 'Summary'

Each custom summary requires a second array, that can be empty, listing fields
to be totaled for each section
If there is no matching total data, no total will show

---------

Looking at the Mobile example:

 $mobilepage = array(
  'DATE' => null,
  'RIGS' => null,
  'SUMMARY' => array('Elapsed', 'MHS av', 'Found Blocks=Blks',
                     Accepted', 'Rejected=Rej', 'Utility'),
  'DEVS+NOTIFY' => array('DEVS.Name=Name', 'DEVS.ID=ID', 'DEVS.Status=Status',
                     'DEVS.Temperature=Temp', 'DEVS.MHS av=MHS av',
                     'DEVS.Accepted=Accept', 'DEVS.Rejected=Rej',
                     'DEVS.Utility=Utility', 'NOTIFY.Last Not Well=Not Well'),
  'POOL' => array('POOL', 'Status', 'Accepted', 'Rejected=Rej',
                  'Last Share Time'));

 $mobilesum = array(
  'SUMMARY' => array('MHS av', 'Found Blocks', 'Accepted', 'Rejected',
                   'Utility'),
  'DEVS+NOTIFY' => array('DEVS.MHS av', 'DEVS.Accepted', 'DEVS.Rejected',
                     'DEVS.Utility'),
  'POOL' => array('Accepted', 'Rejected'));

 $customsummarypages = array('Mobile' => array($mobilepage, $mobilesum));

This will show 5 tables (according to $mobilepage)
Each table will have the chosen details for all the rigs specified in $rigs

 DATE
        A single box with the web server's current date and time

 RIGS
        A table of the rigs: description, time, versions etc

 SUMMARY

        This will use the API 'summary' command and show the selected fields:
                Elapsed, MHS av, Found Blocks, Accepted, Rejected and Utility
        However, 'Rejected=Rej' means that the header displayed for the 'Rejected'
        field will be 'Rej', instead of 'Rejected' (to save space)
        Same for 'Found Blocks=Blks' - to save space

 DEVS+NOTIFY

        This will list each of the devices on each rig and display the list of
        fields as shown
        It will also include the 'Last Not Well' field from the 'notify' command
        so you know when the device was last not well

        You will notice that you need to rename each field e.g. 'DEVS.Name=Name'
        since each field name in the join between DEVS and NOTIFY is actually
        section.fieldname, not just fieldname

        The join code automatically adds 2 fields to each GPU device: 'Name' and 'ID'
        They don't exist in the API 'devs' output but I can correctly calculate
        them from the GPU device data
        These two fields are used to join DEVS to NOTIFY i.e. find the NOTIFY
        record that has the same Name and ID as the DEVS record and join them

 POOL

        This will use the API 'pools' command and show the selected fields:
                POOL, Status, Accepted, Rejected, Last Share Time
        Again, I renamed the 'Rejected' field using 'Rejected=Rej', to save space

$mobilesum lists the sections and fields that should have a total
```

```
You can't define them for 'DATE' or 'RIGS' since they are hard coded tables\
The example given:

 SUMMARY
        Show a total at the bottom of the columns for:
                MHS av, Found Blocks, Accepted, Rejected, Utility

        Firstly note that you use the original name i.e. for 'Rejected=Rej'
        you use 'Rejected', not 'Rej' and not 'Rejected=Rej'

        Secondly note that it simply adds up the fields
        If you ask for a total of a string field you will get the numerical
        sum of the string data

 DEVS+NOTIFY

        Simply note in this join example that you must use the original field
        names which are section.fieldname, not just fieldname

 POOL
        Show a total at the bottom of the columns for:
                Accepted and Rejected

        Again remember to use the original field name 'Rejected'

---------

With cgminer 2.10.2 and later, miner.php includes an extension to
the custom pages that allows you to apply SQL style commands to
the data: where, group, and having
cgminer 3.4.2 and later also includes another option 'gen'
cgminer 4.2.0 and later also includes another option 'fmt'
cgminer 4.2.1 and later also includes another option 'bgen'

An example of an 'ext' section in a more complex custom summary page:

$poolsext = array(
 'POOL+STATS' => array(
        'where' => null,
        'group' => array('POOL.URL', 'POOL.Has Stratum',
                         'POOL.Stratum Active', 'POOL.Has GBT'),
        'calc' => array('POOL.Difficulty Accepted' => 'sum',
                        'POOL.Difficulty Rejected' => 'sum',
                        'STATS.Times Sent' => 'sum',
                        'STATS.Bytes Sent' => 'sum',
                        'STATS.Times Recv' => 'sum',
                        'STATS.Bytes Recv' => 'sum'),
        'gen' => array('AvShr', 'POOL.Difficulty Accepted/max(POOL.Accepted,1)),
        'having' => array(array('STATS.Bytes Recv', '>', 0)),
        'fmt' => 'myfmtfunc'));

function myfmtfunc($section, $name, $value, $when, $alldata,
                   $warnclass, $errorclass, $hiclass, $loclass, $totclass);
{
 $ret = '';
 $class = '';
 switch ($section.'.'.$name)
 {
 case 'GEN.AvShr':
        $ret = number_format((float)$value, 2);
        if ($value == 0)
                $class = $errorclass;
        break;
 // Nonsence example :) since total would show the sum of the averages
 case 'total.AvShr':
        $ret = $value;
        if ($value == 0)
                $class = $warnclass;
        break;
 }
 return array($ret, $class);
}

This allows you to group records together from one or more rigs
In the example, you'll get each Pool (with the same URL+Stratum+GBT settings)
listed once for all rigs and a sum of each of the fields listed in 'calc'


'where' and 'having' are an array of fields and restrictions to apply

In the above example, it will only display the rows where it contains the
'STATS.Bytes Recv' field with a value greater than zero
If the row doesn't have the field, it will always be included
All restrictions must be true in order for the row to be included
Any restiction that is invalid or unknown is true
An empty array, or null, means there are no restrictions

A restriction is formatted as: array('Field', 'restriction', 'value')
Field is the simple field name as normally displayed, or SECTION.Field
if it is a joined section (as in this case 'POOL+STATS')
```

```
The list of restrictions are:                                                \
'set' - true if the row contains the 'Field' ('value' is not required or used)
'=', '<', '<=', '>', '>' - a numerical comparison
'eq', 'lt', 'le', 'gt', 'ge' - a case insensitive string comparison

You can have multiple restrictions on a 'Field' - but all must be true to
include the row containing the 'Field'
e.g. a number range between 0 and 10 would be:
array('STATS.Bytes Recv', '>', 0), array('STATS.Bytes Recv', '<', 10)

The difference between 'where' and 'having' is that 'where' is applied to the
data before grouping it and 'having' is applied to the data after grouping it
- otherwise they work the same


'group' lists the fields to group over and 'calc' lists the function to apply
to other fields that are not part of 'group'

You can only see fields listed in 'group' and 'calc'

A 'calc' is formatted as: 'Field' => 'function'
The current list of operations available for 'calc' are:
'sum', 'avg', 'min', 'max', 'lo', 'hi', 'count', 'any'
The first 4 are as expected - the numerical sum, average, minimum or maximum
'lo' is the first string of the list, sorted ignoring case
'hi' is the last string of the list, sorted ignoring case
'count' is the number of rows in the section specified in the calc e.g.
 ('DEVS.Name' => 'count') would be the number of DEVS selected in the 'where'
 of course any valid 'DEVS.Xyz' would give the same 'count' value
'any' is effectively random: the field value in the 1st row of the grouped data
An unrecognised 'function' uses 'any'


A 'fmt' allows you to specify a function to be called by miner.php to format
data to be displayed in the output html
If the function doesn't exist in miner.php or myminer.php, then it will be
ignored
If the function returns a $ret value (see the example 'myfmtfunc' above) then
that will be displayed, however if $ret is empty, then the normal formatting
code will process the data to be displayed
Thus, if there is no formatting code in miner.php for the field value, then it
will be displayed as it was received from the API
i.e. this allows you to either supply some php code to format field values
that are not formatted by miner.php, or you can also override the formatting
done by miner.php itself for your chosen list of field data
You can return an ' ' if you wish to force it to display as blank
Use the example 'myfmtfunc' above as a template to write your own
Note that your provided function will be called for all data being displayed,
so you should use the 'case' layout as in the example to select the data fields
you wish to format, but return '' for fields you don't wish to change the way
they are formatted
The 2nd return field is the name of a CSS class in $colourtable or created in
your own $add_css_names and $colouroverride
The value you return can stay in effect even if you return an empty $ret, if
the default formatting function for the field doesn't set the $class variable
The fields passed to your function by miner.php:
 $warnclass, $errorclass, $hiclass, $loclass, $totclass
contain the default class names used for formatting


A 'gen' or 'bgen' allows you to generate new fields from any php valid function
of any of the other fields
 e.g. 'gen' => array('AvShr', 'POOL.Difficulty Accepted/max(POOL.Accepted,1)),
will generate a new field called GEN.AvShr that is the function shown, which
in this case is the average difficulty of each share submitted

The difference between 'bgen' and 'gen' is that 'bgen' is done before doing
the 'group' and 'calc', however 'gen' is done after doing 'group' and 'calc'
This means that 'group' and 'calc' can also use 'bgen' fields
As before, 'gen' fields act on the results of the 'group' and 'calc'
If there is no 'group' or 'calc' then they both will produce the same results
Note that 'gen' fields are called 'GEN.field' and 'bgen' fields, 'BGEN.field'

THERE IS A SECURITY RISK WITH HOW GEN/BGEN WORKS
It simply replaces all the variables with their values and then requests PHP
to execute the formula - thus if a field value returned from a cgminer API
request contained PHP code, it could be executed by your web server
Of course cgminer doesn't do this, but if you do not control the cgminer that
returns the data in the API calls, someone could modify cgminer to return a
PHP string in a field you use in 'gen' or 'bgen'
Thus use 'gen' and 'bgen' at your own risk
If someone feels the urge to write a mathematical interpreter in PHP to get
around this risk, feel free to write one and submit it to the API author for
consideration
```

EXHIBIT D

REDACTED IN ITS ENTIRETY

EXHIBIT E

REDACTED IN ITS ENTIRETY

EXHIBIT F

REDACTED IN ITS ENTIRETY

EXHIBIT G

REDACTED IN ITS ENTIRETY

EXHIBIT H

REDACTED IN ITS ENTIRETY

EXHIBIT I

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 July 2019 (18.07.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/139632 A1

(51) International Patent Classification:
G06F 1/26 (2006.01) *G06F 1/30* (2006.01)
G06F 1/20 (2006.01) *G06F 1/32* (2006.01)

(21) International Application Number:
PCT/US2018/017950

(22) International Filing Date:
13 February 2018 (13.02.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/616,348 11 January 2018 (11.01.2018) US

(71) Applicant: **LANCIUM LLC** [US/US]; c/o Angelo Mikeska PLLC, 21 Waterway Avenue, Suite 300, The Woodlands, TX 77380 (US).

(72) Inventors: **HENSON, David**; Lancium LLC, c/o Angelo Mikeska PLLC, 21 Waterway Avenue, Suite 300, The Woodlands, TX 77380 (US). **MCNAMARA, Michael**; Lancium LLC, c/o Angelo Mikeska PLLC, 21 Waterway Avenue, Suite 300, The Woodlands, TX 77380 (US). **CLINE, Raymond**; Lancium LLC, c/o Angelo Mikeska PLLC, 21 Waterway Avenue, Suite 300, The Woodlands, TX 77380 (US).

(74) Agent: **KORENCHAN, James, L.**; McDonnell Boehnen Hulbert & Berghoff LLP, 300 South Wacker Drive, Suite 3200, Chicago, IL 60606 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,

(54) Title: METHOD AND SYSTEM FOR DYNAMIC POWER DELIVERY TO A FLEXIBLE DATACENTER USING UNUTILIZED ENERGY SOURCES

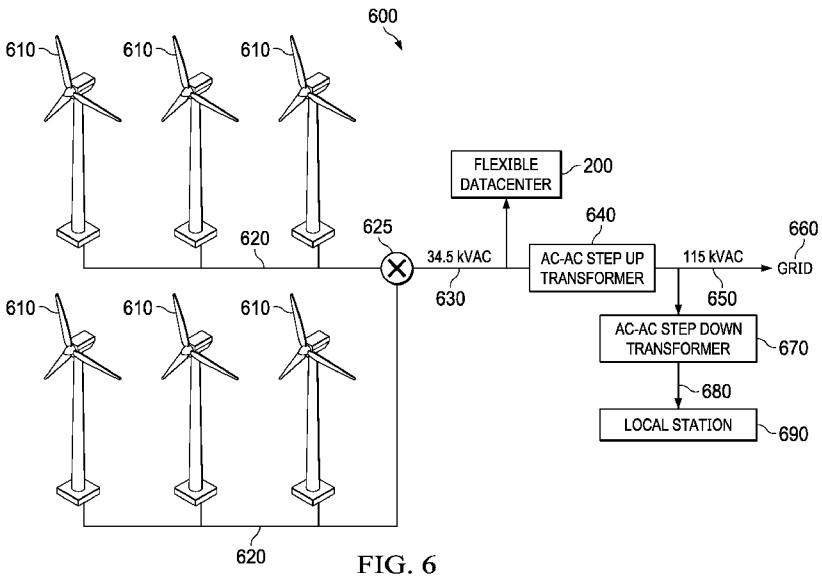


FIG. 6

(57) Abstract: A flexible datacenter includes a mobile container, a behind-the-meter power input system, a power distribution system, a datacenter control system, a plurality of computing systems, and a climate control system. The datacenter control system modulates power delivery to the plurality of computing systems based on unutilized behind-the-meter power availability or an operational directive. A method of dynamic power delivery to a flexible datacenter using unutilized behind-the-meter power includes monitoring unutilized behind-the-meter power availability, determining when a datacenter ramp-up condition is met, enabling behind-the-meter power delivery to one or more computing systems when the datacenter ramp-up condition is met, and directing the one or more computing systems to perform predetermined computational operations.

WO 2019/139632 A1

[Continued on next page]

or number of wind turbines **610**, the configuration or design of wind farm **600**, and grid **660** that it feeds into.

[0050] The output side of AC-to-AC step-up transformer **640** that connects to grid **660** may be metered and is typically subject to transmission and distribution costs. In contrast, power consumed on the input side of AC-to-AC step-up transformer **640** may be considered behind-the-meter and is typically not subject to transmission and distribution costs. As such, one or more flexible datacenters **200** may be powered by three-phase wind-generated AC voltage **620**. Specifically, in wind farm **600** applications, the three-phase behind-the-meter AC voltage used to power flexible datacenter **200** may be three-phase wind-generated AC voltage **620**. As such, flexible datacenter **200** may reside behind-the-meter, avoid transmission and distribution costs, and may be dynamically powered when unutilized behind-the-meter power is available.

[0051] Unutilized behind-the-meter power availability may occur when there is excess local power generation. In high wind conditions, wind farm **600** may generate more power than, for example, AC-to-AC step-up transformer **640** is rated for. In such situations, wind farm **600** may have to take steps to protect its equipment from damage, which may include taking one or more turbines **610** offline or shunting their voltage to dummy loads or ground. Advantageously, one or more flexible datacenters **200** may be used to consume power on the input side of AC-to-AC step-up transformer **640**, thereby allowing wind farm **600** to operate equipment within operating ranges while flexible datacenter **200** receives behind-the-meter power without transmission or distribution costs. The local station control system (not independently illustrated) of local station **690** may issue an operational directive to the one or more flexible datacenters **200** or to the remote master control system (**420** of Figure **4**) to ramp-up to the desired power consumption level. When the operational directive requires the cooperative action of multiple flexible datacenters **200**, the remote mater control system (**420** of Figure **4**) may determine how to power each individual flexible datacenter **200** in accordance with the operational directive or provide an override to each flexible datacenter **200**.

[0052] Another example of unutilized behind-the-meter power availability is when grid **660** cannot, for whatever reason, take the power being produced by wind farm **600**. In such situations, wind farm **600** may have to take one or more turbines **610** offline or shunt their voltage to dummy loads or ground. Advantageously, one or

more flexible datacenters **200** may be used to consume power on the input side of AC-to-AC step-up transformer **640**, thereby allowing wind farm **600** to either produce power to grid **660** at a lower level or shut down transformer **640** entirely while flexible datacenter **200** receives behind-the-meter power without transmission or distribution costs. The local station control system (not independently illustrated) of local station **690** or the grid operator (not independently illustrated) of grid **660** may issue an operational directive to the one or more flexible datacenters **200** or to the remote master control system (**420** of Figure **4**) to ramp-up to the desired power consumption level. When the operational directive requires the cooperative action of multiple flexible datacenters **200**, the remote master control system (**420** of Figure **4**) may determine how to power each individual flexible datacenter **200** in accordance with the operational directive or provide an override to each flexible datacenter **200**.

[0053] Another example of unutilized behind-the-meter power availability is when wind farm **600** is selling power to grid **660** at a negative price that is offset by a production tax credit. In certain circumstances, the value of the production tax credit may exceed the price wind farm **600** would have to pay to grid **660** to offload their generated power. Advantageously, one or more flexible datacenters **200** may be used to consume power behind-the-meter, thereby allowing wind farm **600** to produce and obtain the production tax credit, but sell less power to grid **660** at the negative price. The local station control system (not independently illustrated) of local station **690** may issue an operational directive to the one or more flexible datacenters **200** or to the remote master control system (**420** of Figure **4**) to ramp-up to the desired power consumption level. When the operational directive requires the cooperative action of multiple flexible datacenter **200**, the remote master control system (**420** of Figure **4**) may determine how to power each individual flexible datacenter **200** in accordance with the operational directive or provide an override to each flexible datacenter **200**.

[0054] Another example of unutilized behind-the-meter power availability is when wind farm **600** is selling power to grid **660** at a negative price because grid **660** is oversupplied or is instructed to stand down and stop producing altogether. The grid operator (not independently illustrated) may select certain power generation stations to go offline and stop producing power to grid **660**. Advantageously, one or more flexible datacenters **200** may be used to consume power behind-the-meter, thereby

EXHIBIT J

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX LLC and AUSTIN STORMS,	)	
	)	
Plaintiffs,	)	
	)	
v.	)	C.A. No. 21-534-MN-CJB
	)	
LANCIUM LLC, MICHAEL T. MCNAMARA, and RAYMOND E. CLINE, JR.	)	
	)	
	)	
Defendants.	)	

EXPERT REPORT OF MARK EHSANI, Ph. D.

OUTSIDE COUNSEL'S EYES ONLY - SUBJECT TO PROTECTIVE ORDER

43. I note that Dr. McClellan did not perform any claim construction. Instead, he applied the "plain and ordinary meaning of the claim terms." (McClellan Report, at ¶ 49). But Dr. McClellan did not provide his understanding of the "plain and ordinary meaning" for any claim terms. I understand that Plaintiffs have the burden of proof on the inventorship issue.

44. To determine conception, I also apply the plain and ordinary meaning of the claim terms as would have been understood by a POSITA and, for certain terms, I discuss the specification's use of those terms. I reserve the right to supplement my report should Plaintiffs' or any of Plaintiffs' experts use a different construction, or provide their understanding of the plain and ordinary meaning that is different than my understanding of the plain and ordinary meaning.

VII. LANCIUM INDEPENDENTLY DEVELOPED, CONCEIVED, AND REDUCED ITS TECHNOLOGY TO PRACTICE, INCLUDING EACH OF THE INVENTIONS CLAIMED IN THE '433 PATENT

45. It is my opinion that Lancium independently developed, conceived, and reduced its technology to practice, including each of the inventions claimed in the '433 patent, and that such development, conception, and reduction to practice did not involve the use of any information allegedly provided to Mr. McNamara by Mr. Storms. As discussed above, my opinions are based on, among other things, my review of: (1) pleadings; (2) the '632 application and the '433 patent and its file history; (3) the parties' Responses to Discovery (including documents cited in those responses); (4) the deposition testimony from this case and exhibits cited in same; (5) my review of other documents and materials; (6) the communications between Mr. Storms and Mr. McNamara; and (7) my education and forty-plus years of experience. I also relied in part on Mr. Siddiqi's analysis as set forth in his report, and Mr. Baer's analysis of the source code produced by Mr. Storms and relied upon by Dr. McClellan as discussed in Mr. Baer's report.

Under Frequency Relay (UFR) that can trip the NCLR if the grid frequency drops below 59.7 Hz[28] with a delay time of 20 cycles, among other qualifications. The UFR is active and available to automatically trip the NCLR during each of the award hours. Additionally, the NCLR can be tripped manually by ERCOT instruction during each of the award hours.[29] Actually tripping the NCLR is referred to as "deploying" the NCLR – the deployment causes an electrical breaker to trip and removes the load from the grid, cutting the supply of grid power to the NCLR. As part of the obligation created upon receiving an award from the ancillary service DAM market, an NCLR participating in RRS must consume at least the awarded quantity of energy for each awarded hour in order to fulfill their obligation to be able to drop the awarded load upon deployment.[30]

59. A drawback to bidding into the NCLR RRS market is that the market is typically oversubscribed. Though an NCLR may bid its preferred quantities of energy into the NCLR RRS market for each hour in the DAM, the actual awarded energy for each hour will typically be less than the bid because there are more market participant bids than available capacity in the NCLR RRS market.[31] For example an NCLR with a 50MW load capacity may bid the full 50MW into the NCLR RRS market for each hour in hour 1 through hour 8 in the DAM. After DAM clearance, the award to the NCLR may only be, as an example, 20MW for each hour in hour 1 through hour 8. Thus the NCLR will receive ancillary service settlement payments at the settlement price for each hour for 20MW. However, if the NCLR is actually deployed (either by UFR under-frequency

[28] Grid frequency in the U.S. is nominally 60 Hertz ("Hz"). If generation and load are unbalanced on the grid, the frequency deviates from 60 Hz. If the grid frequency drops below the 60 Hz, then that is an indication that load exceeds generation.

[29] *See*, *e.g.*, Siddiqi Report, Section 6.6

[30] *See*, *e.g.*, Siddiqi Report, Section 6.6

[31] *See*, *e.g.*, Siddiqi Report, Section 3.4.

trip or manual instruction from ERCOT), then the breaker will trip and remove all power from the NCLR. While the NCLR is only obligated under the award to consume at least 20MW for each hour between hour 1 and hour 8, the NCLR might actually be operating at its full 50MW. Under that scenario, the NCLR would lose its full 50MW of operation, but would only be compensated via ancillary services settlement for the 20MW awarded.[32]

60. A CLR could theoretically bid into CLR RRS and CLR Regulation Up Service within the ERCOT ancillary services market, among other services.[33] However, CLR qualification is significantly more rigorous than NCLR qualification. It is my understanding that as of 2019, no Load Resource had ever qualified as a CLR. When Lancium qualified as a CLR in June 2020, it is my understanding that Lancium became the first Load Resource in ERCOT to ever qualify as a CLR.[34] Similar to an NCLR, a CLR participating in the CLR RRS or CLR Reg-Up programs are subject to an obligation upon receiving an award from the ancillary service DAM market. A CLR participating in RRS or Reg-Up must consume at least the awarded quantity of energy for each awarded hour in order to fulfill their obligation to be able to drop the awarded load upon dispatch.[35] A CLR may be dispatched by ERCOT at any time during the award hours.

61. As of the date of this report a CLR does not experience the same drawbacks to bidding into the CLR RRS and Reg-Up markets as an NCLR experience bidding into the NCLR RRS market. According to Dr. Siddiqi, there are now only eight CLRs currently registered in

[32] Some of the operational revenue the NCLR may have generated with this lost capacity might be mitigated by power sell-back if the real-time market (discussed below) has high prices and the NCLR has previously purchased power that it can sell back in the real-time market.

[33] *See*, *e.g.*, Siddiqi Report, Section 6.5

[34] The Siddiqi Report (Section 6.5) identifies some of the qualifications for a CLR.

[35] *See*, *e.g.*, Siddiqi Report, Section 6.5.

76. I understand and the documentary evidence supports that through June, July, and August 2019, Lancium and MP2 worked toward the goal of installing the required metering and UFR for enrolling Lancium in the ERCOT ERS program. During this period, Lancium and MP2 began to appreciate that Lancium fast ramping control could also allow them to participate in non-ERS Load Resource (i.e., ancillary service) programs such as NCLR. By July 15, 2019, Lancium and MP2 had fully executed MP2's standard agreement for ERS and Load Resource services.[49] On August 22, 2019, Lancium first participated as a Load Resource in ERCOT via MP2.[50] Lancium initially did not fully understand how the bids, awards, and revenue worked for such participation, but MP2 explained the process on August 27.[51] By August 28, 2019, Lancium had adjusted its economic curtailment performance strategies in their control software to make sure they consumed the obligated load they were awarded each day.[52] Also, during this period, Lancium began to appreciate the economic advantage of applying their software control of load to take advantage of traditional energy arbitrage. For example, on August 14, 2019, Lancium moved to a standard fixed-price energy contract with Calpine.[53] Thereafter, instead of their prior practice of using the Lancium software to ramp up and down based on avoiding high power prices, Lancium

[49] LANCIUM00030572; LANCIUM00030573

[50] LANCIUM00033062;

[51] LANCIUM00024122; LANCIUM00033240; LANCIUM00030839; LANCIUM00030840; LANCIUM00030841; LANCIUM00030842

[52] LANCIUM00024173.

[53] LANCIUM00028482

used their software to ramp their grid-connected miners up and down based on RTM prices, thereby obtaining additional revenue from power sell-back during periods of high RTM prices.[54]

77. I understand and the documentary evidence supports that in September 2019, Lancium, while at the offices of MP2, first fully demonstrated their fast-ramping technology to MP2.[55] There was an immediate realization following the demonstration that Lancium could potentially use their technology to qualify for a new revenue source for their grid-connected miners: CLR.[56] Lancium's previously developed fast-ramping control software was able to ramp their grid-connected loads sufficiently fast to allow Lancium to meet the rigorous qualifications required of a CLR.[57] Thus far, no other entity had ever qualified as a load-only CLR.[58] Lancium quickly began what would become a nearly year-long process of engaging with MP2 and ERCOT to become the first ever qualified load-only CLR in June 2020.[59]

78. In the sections below, I provide a more detailed review of Lancium's technology development efforts.

---

[54] LANCIUM00033064; LANCIUM00033065

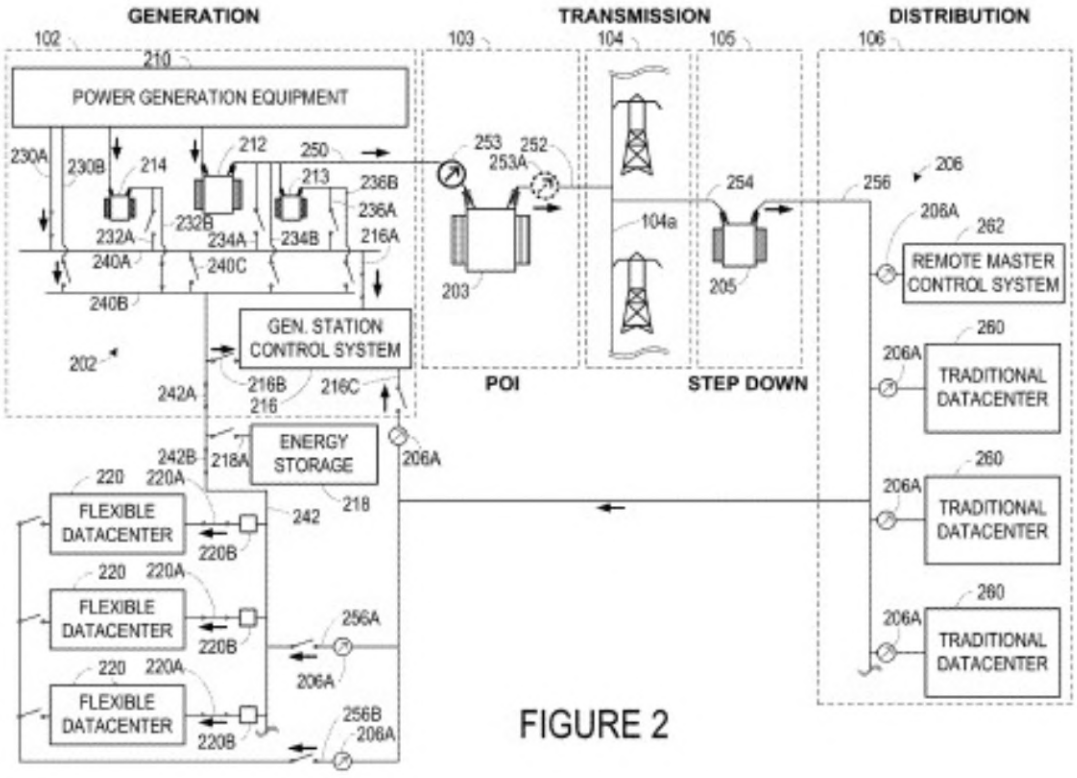
[55] LANCIUM00024131.

[56] LANCIUM00021587.

[57] *See, e.g.,* LANCIUM00030797.

[58] LANCIUM00021587.

[59] https://www.prnewswire.com/news-releases/lancium-and-mp2-energy-offer-unique-energy-demand-response-solution-for-high-throughput-computing-and-cryptocurrency-miners-301080410.html.



106. The '433 patent also discloses and claims a control system for controlling the flexible datacenters. Referring to Figure 11 (below), a system for implementing control strategies based on a power option agreement is disclosed. The system 1100 is an example arrangement that includes a control system (*e.g.*, the remote master control system 262 of Figs. 2 and 3), a load (*e.g.*, one or more datacenters 1102, 1104, 1106), and a power entity 1140.[100]

[100] '433 patent, at 43:35-45.

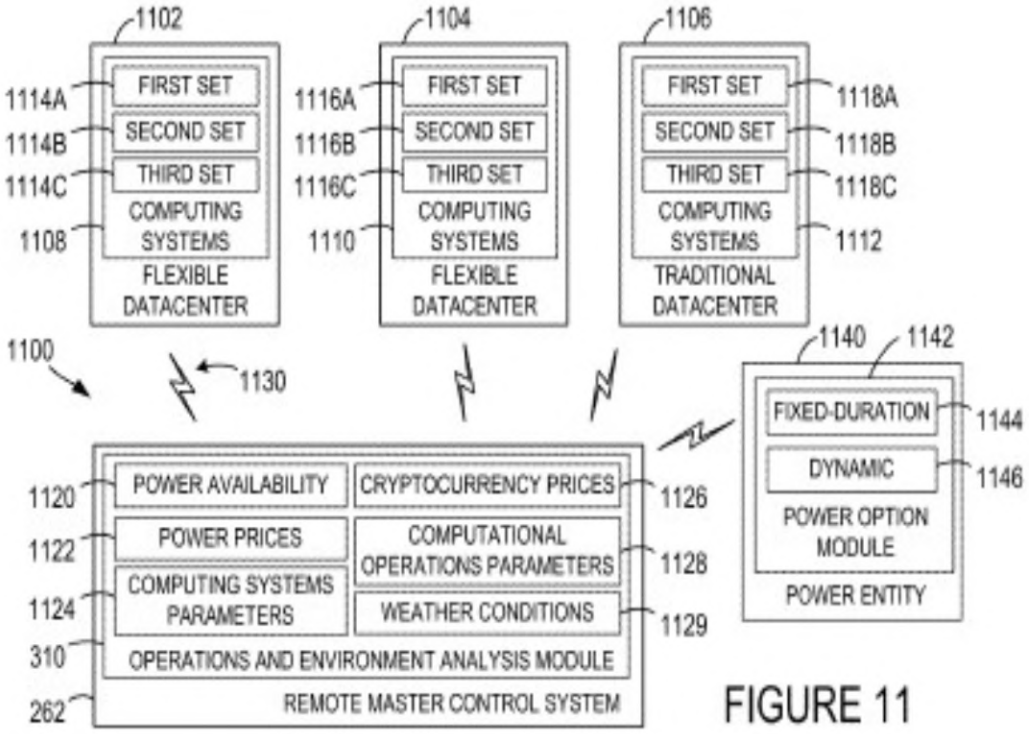


FIGURE 11

107. The '433 patent specification describes a "power option agreement" as an agreement between a power entity associated with the delivery of power (*e.g.*, a grid operator, power generation station, or local control station) and the load (*e.g.*, a datacenter such as 1102).[101] As part of the power option agreement, the load provides the power entity with the right, but not the obligation, to reduce the amount of power delivered (*e.g.*, grid power) to the load up to an agreed amount of power during an agreed upon time interval.[102] To provide the power entity this option, the load needs to be using at least the amount of power subject to the option (*e.g.*, the minimum power threshold),[103] and may grant the power entity with this option in exchange for

[101] *Id.* at 43:46-50.

[102] *Id.* at 43:50-57.

[103] *Id.* at 43:57-65.

d) Element 1[b][2] – "receive power option data based, at least in part, on a power option agreement, wherein the power option data specify: (i) a set of minimum power thresholds, and (ii) a set of time intervals, wherein each minimum power threshold in the set of minimum power thresholds is associated with a time interval in the set of time intervals;"

115. In my opinion, with respect to the specific claims of the '433 patent, Messrs. McNamara and/or Cline conceived this element independently and without utilization of any information allegedly provided by Mr. Storms. As described above and in SSR to ROG 3, by March 2018 McNamara and/or Cline understood that Lancium's flexible datacenters could ramp to absorb and drop power within five minute windows and that these datacenters could be operated remotely via Lancium's Network Operations Center ("NOC") that could respond to signals from grid operators.[120] Messrs. McNamara and/or Cline Lancium were, at this time, also aware of ERCOT and at least peripherally aware of ancillary services.[121] Messrs/ McNamara and/or Cline continued to develop this technology and by October 2018 had spent more than $1M on R&D infrastructure, container design, and software development.[122]

116. Messrs. McNamara, Cline, and Lancium continued to develop their technology through 2019,[123] but it was not until the Summer and Fall of 2019 that Messrs. McNamara and Cline appreciated the benefits of applying their technology to ancillary serves and that is when they subsequently conceived of using their technology for receiving power option data based on a power option agreement specifying a set of minimum power thresholds and a set of time intervals

---

[120] *See*, *e.g.*, SSR to ROG 3, at 16 and documents cited therein.

[121] *Id*.

[122] SSR to ROG 3, at 19 and documents cited therein, including LANCIUM00021489 for R&D numbers.

[123] *See, e.g.,* SSR to ROG 3, at 19-28 (and documents cited therein).

in furtherance of performing ancillary services with their fast-ramping datacenters. On August 27, 2019, Mr. Cline sent an email to Mr. McNamara noting an important point that had not come across in previous conversations with their QSE: the "award" (received after offering into an ancillary services program and received as part of the establishment of the power option agreement) is essentially an "***obligation on our [Lancium's] part, that we consume that amount of power the ERCOT COULD curtail***."[124] In my opinion, this was the flash of insight that led to Messrs. McNamara and Cline conception, at least because this is when they first understood that their system would need to receive the award (the received set of power option data) which would be sent to Lancium based on, and in response to, the accepted prior offer by Lancium (which forms the awarded power option agreement) and the award includes the set of MW (*i.e.*, the set of ***minimum*** power thresholds) corresponding to each hour of the awarded offer (*i.e.*, the set of time intervals).[125] In further support of my opinion, I note that on August 28, 2019, Mr. Cline sends an email to MP2 confirming on Lancium's behalf that, "[w]e are adjusting our economic curtailment plans to assure that we consume the obligated load we have been awarded. If we go below that level we will coordinate with the operations desk. We understand that we cannot receive an award for power that could be curtailed, if we are not using the power."[126] I believe that this correspondence indicates the approximate date that Mr. Cline and Mr. McNamara formed in their

[124] *See* LANCIUM00030839; *see also* SSR to ROG 3, at 29-31 (and documents cited therein).

[125] LANCIUM00030799 and LANCIUM00030801 (examples of awards); *see also* Siddiqi Report, Section 6.4 ("No later than 1:30 in the Day-Ahead, ERCOT shall notify the parties to each cleared DAM transaction (*e.g.*, the buyer and seller) of the results of the DAM including awarded Ancillary Service Offers, specifying Resource, MW, Ancillary Service type, and price, for each hour of the awarded offer").

[126] *See* LANCIUM00024173; *see also* SSR to ROG 3, at 29-31 (and documents cited therein).

minds the definite and permanent idea claimed in this limitation and the following limitations of Claim 1.

> **e) Element 1[b3] – "responsive to receiving the power option data, determining a performance strategy for the set of computing systems based on a combination of at least a portion of the power option data and at least one condition in the set of conditions, wherein the performance strategy comprises a power consumption target for the set of computing systems for each time interval in the set of time intervals, wherein each power consumption target is equal to or greater than the minimum power threshold associated with each time interval; and"**

117. In my opinion, with respect to the specific claims of the '433 patent, Messrs. McNamara and Cline conceived this element independently and without utilization any information allegedly provided by Mr. Storms. I refer back to my analysis immediately above that on or about August 28, 2019, Mr. Cline indicated he understood that the award was an obligation for their Lancium's flexible datacenter system to operate above a ***minimum*** threshold load (*e.g.*, the awarded MW) for each time interval (each hour in the award). As described previously and in other places in my report, Messrs. McNamara and Cline had previously developed control systems for its flexible datacenters that had long monitored or considered other various conditions and developed strategies based on those conditions, including economic considerations such as price response, energy arbitrage, and consideration of the profitability of mining. Following the flash of insight, it is my opinion that Messrs. Cline and/or McNamara then understood that Lancium's control system would need to determine the performance strategy described in this limitation and now adjust a power consumption target for the load by determining a power consumption target for the set of computing systems for each time interval that was ***either equal to or greater than***, the ***minimum*** power threshold associated with each time interval in the award. For example, if economic conditions were favorable for mining, the control system would set the power

I declare under the penalty of perjury under the laws of the United States that the foregoing is true and correct.

Executed this 6th day of May, 2022, in College Station, Texas.

Mark Ehsani, Ph.D., P.E., LF. IEEE, F. SAE, M. AAAS

EXHIBIT K

REDACTED IN ITS ENTIRETY

EXHIBIT L

REDACTED IN ITS ENTIRETY

EXHIBIT M

From: Jay Young <Jyoung@strategicpowersolutions.net>
To: Todd Wilson <Todd.Wilson@calpinesolutions.com>, Michael McNamara <michael.mcnamara@lancium.com>
CC: Jon Cohen <jon.cohen@lancium.com>
Subject: Re: ERS
Date: Wed, 15 May 2019 21:54:06 +0000

An invite came from my assistant Brooke Bassler. Please check spam if you did not receive.

Thank you,

Jay Young
214-415-5462

From: Todd Wilson <Todd.Wilson@calpinesolutions.com>
Date: Wednesday, May 15, 2019 at 6:37 AM
To: Michael McNamara <michael.mcnamara@lancium.com>, Jay Young <Jyoung@strategicpowersolutions.net>
Cc: Jon Cohen <jon.cohen@lancium.com>
Subject: RE: ERS

Good by me

Best Regards,

Todd

Todd Wilson
Cell: (919) 414-7986

From: Michael McNamara [mailto:michael.mcnamara@lancium.com]
Sent: Tuesday, May 14, 2019 10:04 PM
To: Jay Young <Jyoung@strategicpowersolutions.net>
Cc: Jon Cohen <jon.cohen@lancium.com>; Todd Wilson <Todd.Wilson@calpinesolutions.com>
Subject: Re: ERS

External Sender: Use caution with links/attachments.

1 central would be great.

On Tue, May 14, 2019 at 10:02 PM Jay Young <Jyoung@strategicpowersolutions.net> wrote:

> Send me a time and I will have a conference line set and sent out.
>
> Sent from my iPhone
>
> On May 14, 2019, at 10:00 PM, Michael McNamara <michael.mcnamara@lancium.com> wrote:
>
> Thursday is great for me. Could you please send a calendar invite?
>
> On Tue, May 14, 2019 at 2:47 PM Jay Young <Jyoung@strategicpowersolutions.net> wrote:

How does Thursday look for everyone? I'm open most of the day until 2pm central.

Jay Young
214-415-5462

From: Jon Cohen <jon.cohen@lancium.com>
Date: Tuesday, May 14, 2019 at 2:45 PM
To: Todd Wilson <Todd.Wilson@calpinesolutions.com>
Cc: Jay Young <Jyoung@strategicpowersolutions.net>, Michael McNamara <michael.mcnamara@lancium.com>
Subject: Re: ERS

Thanks,
So seems like no times work tomorrow for both Jay and Todd.

Michael's on the west coast so if cares to join, afternoon is better. Otherwise I'm free in the morning

On Tue, May 14, 2019 at 12:13 PM Todd Wilson <Todd.Wilson@calpinesolutions.com> wrote:

> I am available any time after 9:45 up till 2:30
>
> Best Regards,
>
> Todd
>
> Todd Wilson
> Cell: (919) 414-7986
>
> **From:** Jon Cohen [mailto:jon.cohen@lancium.com]
> **Sent:** Tuesday, May 14, 2019 10:26 AM
> **To:** Todd Wilson <Todd.Wilson@calpinesolutions.com>
> **Cc:** Michael McNamara <michael.mcnamara@lancium.com>; Jay Young <jyoung@strategicpowersolutions.net>
> **Subject:** Re: ERS
>
> **External Sender: Use caution with links/attachments.**
>
> Thanks Todd,
> Jay, Michael and I are tied up today, but would appreciate the opportunity to chat tomorrow if you're around?
> Jon
>
> On Tue, May 14, 2019 at 10:29 AM Todd Wilson <Todd.Wilson@calpinesolutions.com> wrote:
>
> > Gentlemen,
> >
> > As you may know, Calpine Energy Solutions decided some years back that we were going to focus solely on risk management of our client's power portfolio. To that end we have partnered with other organizations that are similarly focused in their respectful fields.
> >
> > To that end, I would like to introduce you to Jay Young who is President of Strategic Power Solutions. I have worked with Jay for over 15 years and he has my utmost respect and confidence. His main focus is the Demand Response programs inside of ERCOT so I believe he will be a great resource for Lancium.

Please expect a call from Jay this morning. I have attached business cards for your respective files.

Best Regards,

Todd

Todd Wilson
Cell: (919) 414-7986

From: Michael McNamara [mailto:michael.mcnamara@lancium.com]
Sent: Monday, May 13, 2019 9:48 AM
To: Jon Cohen <jon.cohen@lancium.com>; Todd Wilson <Todd.Wilson@calpinesolutions.com>
Subject: Re: ERS

External Sender: Use caution with links/attachments.

2MW maybe going up to 6MW

On Mon, May 13, 2019 at 9:10 AM Todd Wilson <Todd.Wilson@calpinesolutions.com> wrote:

> Goof morning Jon, just got back from a mini vacation. How much load were you looking to put into the program?
>
> Best Regards,
>
> Todd
>
> Todd Wilson
> Cell: (919) 414-7986
>
> **From:** Jon Cohen [mailto:jon.cohen@lancium.com]
> **Sent:** Friday, May 10, 2019 1:59 PM
> **To:** Todd Wilson <Todd.Wilson@calpinesolutions.com>; Michael McNamara <michael.mcnamara@lancium.com>
> **Subject:** ERS
>
> **External Sender: Use caution with links/attachments.**
>
> Hi Todd,
> Do you have any intro material on participating in EROT's ERS program? We think our load is well suited, but were curious as to what the process and requirements are.
>
> Also we've been working with Centerpoint on looking at upgrade options for Thomas Rd, but its taking longer than anticipated. Will keep you posted on that
>
> Thanks,
> Jon
>

--

(917) 833-2720
COMPANY CONFIDENTIALITY NOTICE: The information in this e-mail may be confidential and/or privileged and protected by work product immunity or other legal rules. No confidentiality or privilege is waived or lost by mis-transmission. If you are not the intended recipient or an authorized representative of the intended recipient, you are hereby notified that any review, dissemination, or copying of this e-mail and its attachments, if any, or the information contained herein is prohibited. If you have received this e-mail in error, please immediately notify the sender by return e-mail and delete this e-mail from your computer system.

COMPANY CONFIDENTIALITY NOTICE: The information in this e-mail may be confidential and/or privileged and protected by work product immunity or other legal rules. No confidentiality or privilege is waived or lost by mis-transmission. If you are not the intended recipient or an authorized representative of the intended recipient, you are hereby notified that any review, dissemination, or copying of this e-mail and its attachments, if any, or the information contained herein is prohibited. If you have received this e-mail in error, please immediately notify the sender by return e-mail and delete this e-mail from your computer system.

--

(917) 833-2720

--

(917) 833-2720
COMPANY CONFIDENTIALITY NOTICE: The information in this e-mail may be confidential and/or privileged and protected by work product immunity or other legal rules. No confidentiality or privilege is waived or lost by mis-transmission. If you are not the intended recipient or an authorized representative of the intended recipient, you are hereby notified that any review, dissemination, or copying of this e-mail and its attachments, if any, or the information contained herein is prohibited. If you have received this e-mail in error, please immediately notify the sender by return e-mail and delete this e-mail from your computer system.

EXHIBIT N

From: Jay Young <Jyoung@strategicpowersolutions.net>
To: Jon Cohen <jon.cohen@lancium.com>, Michael McNamara <michael.mcnamara@lancium.com>
CC: Todd Wilson <todd.wilson@calpinesolutions.com>
Subject: LR DEMAND RESPONSE PRESENTATION 2019.pptx
Date: Sat, 18 May 2019 14:35:09 +0000
Attachments: LR_DEMAND_RESPONSE_PRESENTATION_2019.pptx

Jon and Michael –

Attached is the information we discussed on Thursday for the DR programs in ERCOT. This covers all of the high level details. We will need to do a site and software assessment to see what is needed for telemetry, communication, etc.

CPower mentioned that the quickest they have enrolled a new participant was 35 days. So if we want to take advantage of historically high summer revenues, we need to get on this next week.

I used to work for CPower and continue to do business with them. They have a great team and are one of the largest QSE's in the nation.

Please let me know if you have any questions.

Thank you,

Jay A. Young
President
Strategic Power Solutions
214-415-5462

EXHIBIT O

REDACTED IN ITS ENTIRETY

EXHIBIT P

REDACTED IN ITS ENTIRETY

EXHIBIT Q

REDACTED IN ITS ENTIRETY

EXHIBIT R

REDACTED IN ITS ENTIRETY

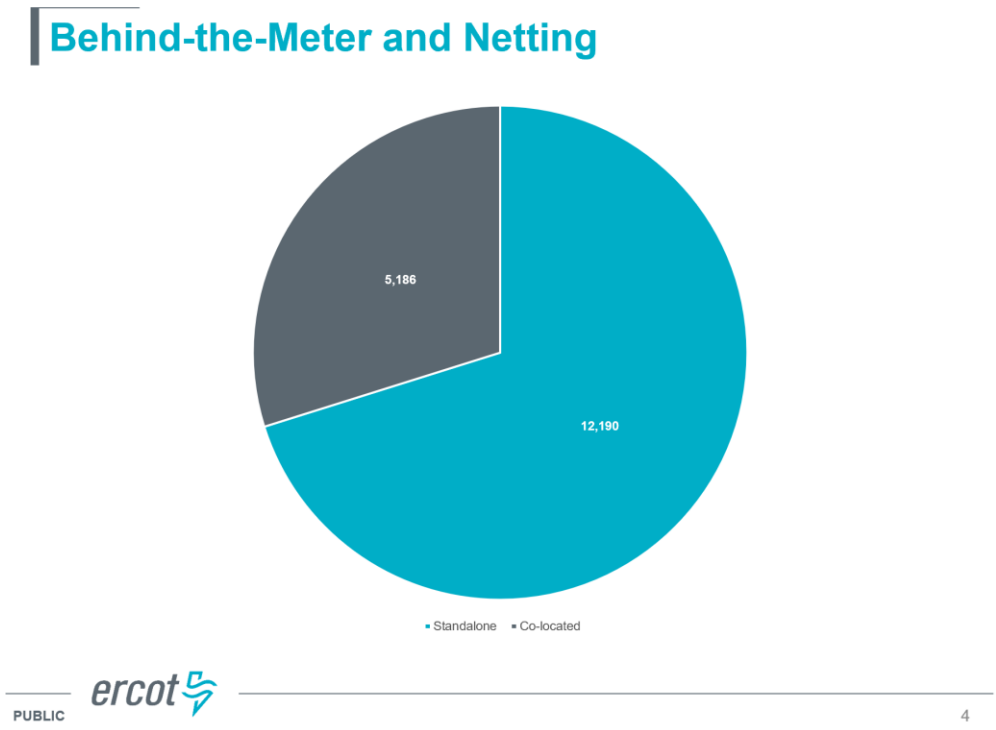
EXHIBIT S

Report

ERCOT Market Mechanisms

Prepared by
Shams Siddiqi, Ph.D.

May 6, 2022



6.1 Load Participation in Energy Market – Price Response

Loads can participate in the ERCOT market by simply reducing consumption during Real-Time Market (RTM) high price intervals. At the request of larger loads, the Load Serving Entity (LSE) serving that load can pass through Four Coincident Peak (4CP) price used to recover transmission cost of service and RTM Settlement Point Prices (SPP) for corresponding Load Zone (LZ). Loads can then voluntarily (i.e., without being instructed by ERCOT) respond (i.e., reduce or curtail load) to avoid 4CP charge and RTM SPPs that are greater than the load's value of consuming that energy (breakeven price) thereby managing their exposure to high (and, from the perspective of the load, uneconomic) energy prices. Large industrial and commercial loads have been responding to prices in this way at the least soon after retail deregulation in 2001.

Controllable Load Resource (CLR), avoiding 4CP intervals, can also submit RTM Energy Bid at the CLR's breakeven price into SCED and thus be dispatched by SCED to maximize its profit.

6.2 Load Participation in Energy Market – Energy Arbitrage

Loads can participate in the ERCOT market by buying energy in advance either bilaterally from the LSE or from the ERCOT-run Day-Ahead Market (DAM) facilitated by the Qualified Scheduling Entity (QSE) for the LSE and then selling back that energy in RTM if the price in the RTM is higher than the load's value of consuming that energy – this is commonly known as

"energy arbitrage." For example, say a load consumes 1 MWh of energy in an hour and purchases that energy from the LSE at $60/MWh and the load has a contractual agreement with the LSE to pay the load the RTM SPP for the corresponding LZ if the load does not consume that energy. If in real-time, the LZ RTM SPP is $300/MWh and the load's value of consuming that energy is $200/MWh (e.g., from cryptocurrency mining activity) then the load voluntarily (i.e., without being instructed by ERCOT) reduces consumption and pays the LSE $60 and is paid by the LSE $300 – thereby net being paid $240 by the LSE. If the load is perfectly flexible, then the load can maximize its profits by curtailing whenever RTM energy prices plus and transmission and distribution charges are greater than the value the load produces by consuming that energy.

In a similar way, with the help of the QSE, the load could purchase 1 MWh of energy for that hour in the DAM at say $40/MWh and the load has a contractual agreement with the LSE to pay the load the RTM SPP for the corresponding LZ if the load does not consume that energy. If in real-time, the LZ RTM SPP is $500/MWh and the load voluntarily (i.e., without being instructed by ERCOT) reduces consumption if this price is above the value the load derives from consuming that energy, then the load will pay the LSE $40 for energy purchased in DAM and be paid by the LSE $500 for energy not consumed in RTM – thereby net being paid $460 by the LSE. The QSE for the LSE essentially bought the energy for $40 in DAM and paid ERCOT for that purchase and then is paid by ERCOT in RTM $500 for that energy that was bought and not consumed (so resold into the RTM). Large industrial and commercial loads have been engaged in "energy arbitrage" in this way at the least soon after retail deregulation in 2001.

Controllable Load Resource (CLR), avoiding 4CP intervals, can also submit RTM Energy Bid at the CLR's breakeven price into SCED and thus be dispatched by SCED to maximize its profit as described in Section 3.1 of this Report.

6.3 Load Participation in Emergency Response Service (ERS)

Emergency Response Service (ERS) is provided by loads and certain generators willing to interrupt or export energy onto the ERCOT system prior to or during an electric grid emergency in exchange for a payment. ERS Resources must be represented by a Qualified Scheduling Entity (QSE). Typically, smaller industrial/commercial loads provide ERS.

ERS resources are procured for one or more of the four ERS service types:
Non-Weather-Sensitive ERS-10 and ERS-30 Weather-Sensitive ERS-10 and ERS-30 (10 and 30 in each type designates the minutes between issuance of ERCOT deployment instruction and full deployment of resource or ramp period)

ERS resources are procured for two 4-month and two 2-month Standard Contract Terms (SCT):
December thru March; April thru May; June thru September; October thru November

Offers may be submitted to provide Emergency Response Service for one or more of the

following Time Periods (TP) in the SCT:
TP1 – Weekdays HE 6-9; TP2 – Weekdays 10-13, TP3 – Weekdays 14-16; TP4 – Weekdays 17-19
TP5 – Weekdays 20-22; TP6 – Weekends 6-9; TP7 – Weekends 16-21; TP8 – all other hours
Note: Weekdays exclude ERCOT Holidays and Weekends include ERCOT Holidays

ERS Procurement constraints are determined using ERS Capacity Demand Curve and Risk Factors to allocate annual spend limit of $50 million. 15-minute interval metering is required for participating in ERS and is used for all availability and performance measurement analysis. ERS may be dispatched when Physical Responsive Capacity (PRC) falls below 3,000 MW and is not projected to be recovered above 3,000 MW within 30 minutes following the deployment of Non-Spin. ERCOT Operations will release ERS resources after the emergency is over via another hotline call. The ERS resource must be back online and ready to perform within 10 hours of the release. When deployed, ERS resources must shed or generate at least 95% of their committed obligation within 10 or 30 minutes, depending on the duration of ERS provided. ERCOT may conduct an unannounced test of any ERS resource at any time during an ERS Time Period in which the ERS resource is contracted to provide ERS. Performance is based on both the availability of the ERS Resource and Test/Event Performance during the SCT. ERS resources are settled using resource-level Test performance factors and the QSE SCT portfolio level Availability and Event performance factors. If the portfolio performs at or above 100%, it will be paid for its entire obligation.

Approximately 1000 MW of ERS resource procured in each of the 8 Time Periods. Typically, smaller industrial/commercial loads but also residential aggregations provide ERS.

6.4 Load Participation in ERCOT Ancillary Service Market

Load Resource is a Load capable of providing Ancillary Service to the ERCOT System and/or energy in the form of Demand response and registered with ERCOT as a Load Resource.

As described in Section 3.4 of this Report, by 10am in Day-Ahead, QSE may submit Load Resource-specific AS Offers for DAM and may offer the same capacity for any or all of those AS products that it is qualified to provide simultaneously. Offers of more than one AS product from one Load Resource may be inclusive or exclusive of each other. No later than 1:30pm in the Day-Ahead, ERCOT shall notify the parties to each cleared DAM transaction (e.g., the buyer and the seller) of the results of the DAM including awarded Ancillary Service Offers, specifying Resource, MW, Ancillary Service type, and price, for each hour of the awarded offer.

There are three types of Load Resources:

Controllable Load Resource (CLR) is a Load Resource capable of controllably reducing or increasing consumption under Dispatch control by ERCOT.

Non-Controllable Load Resource (NCLR) is a Load Resource controlled by high-set under frequency relay (UFR).

Aggregate Load Resource (ALR) is a Load Resource that is an aggregation of individual metered sites, each of which has less than ten MW of Demand response capability and all of which are located within a single Load Zone. There have been no ALR registered and qualified since service was developed.

All Load Resources must meet the following qualifications:

- Need to be registered as Resource Entity (RE) and modeled in ERCOT Systems.
- Need to have revenue grade meter capable of recording 15-minute interval data.
- Data needs to be submitted at least monthly into ERCOT systems.
- Provide a one-line diagram for the Resource including connections all the way to the transmission Load transformer.
- Must be represented by same QSE that represents the Load Serving Entity.
- Need to be qualified to provide Ancillary Services with ERCOT.
- Can have a provisional qualification for 90 days but must eventually pass a test that demonstrates their ability to provide the Ancillary Service.
- Subject to annual unannounced testing.
- Qualifications can be revoked and then suspended for failure to perform during events or tests.
- Telemetry must be validated annually.

6.5 Load Participation as Controllable Load Resource (CLR)

CLRs must meet the following qualifications:

- Provide a test report demonstrating their droop characteristics for CLRs.
- Provide test data to show the CLR is capable of sustaining their Load for at least one hour.
- Pass an initial Security Constrained Economic Dispatch (SCED) qualification test by responding to ERCOT issued basepoint changes that will ramp the CLR between its Maximum Power Consumption (MPC) and Low Power Consumption (LPC), including some holding periods
- Must be capable of following SCED basepoints and, when providing Regulation Service, must be capable of following Load Frequency Control (LFC) deployments.
- Must provide Primary Frequency Response (PFR) similar to the governor for a generator.
- When qualified can provide Responsive Reserve Service (RRS), Regulation Up Service (Reg-Up), Regulation Down Service (Reg-Down), and Non-Spinning Reserve Service (Non-Spin).
- CLR must be represented by the same QSE that represents the Load Serving Entity.
- QSE submits Real-Time Energy Bid representing the energy price above which the CLR will drop Load used by SCED to economically dispatch the CLR – CLR must follow SCED basepoint instructions.
- CLR providing RRS and Reg-Up are dispatched by SCED to consume at least the

amount of RRS and/or Reg-Up provided unless deployed as Ancillary Service. Thus, SCED issues a basepoint instruction to the CLR at the start of its RRS and/or Reg-Up responsibility to consume at least the amount of RRS and Reg-Up provided, regardless of the CLR's Real-Time Energy Bid. When the CLR is no longer providing AS, it is dispatched based on its submitted Real-Time Energy Bid. For system frequency deviations outside the frequency deadband, CLR must provide Primary Frequency Response (PFR) by increasing or decreasing consumption on top of its SCED basepoint. When providing Regulation Service, CLR must additionally respond to 4-second signals from the Load Frequency Control (LFC) to increase or decrease consumption. When providing RRS and RRS is released to SCED when frequency drops below 59.91 Hz, CLR is then dispatched by SCED for its full consumption range based on its submitted Real-Time Energy Bid.

Loads currently participating as CLRs are as follows[5]:

- Large data mining facilities with sophisticated control systems - eight data mining loads totally about 750 MW of controllable load resources.
- Several of these existing CLRs are proposing an additional 2,600 MW pending additional studies by TDSP and ERCOT.
- Seven additional sites have initiated registration (not clear yet if CLR or NCLR).
- The charging loads of Energy Storage Resources are also modeled as CLRs.

6.6 Load Participation as Non-Controllable Load Resource (NCLR)

NCLRs currently consist of blocky loads that deploy by opening a breaker meeting the following qualifications:

- NCLR Under Frequency Relay (UFR) required to trip load for frequency below 59.7 Hz with a delay time of 20 cycles – tested prior to being put in service.
- Required to have UFR when providing Responsive Reserve Service (RRS).
- To provide RRS must have UFR set to trip when frequency drops to 59.7 Hz.
- To provide Non-Spin must have UFR deactivated – cannot provide both RRS and Non-Spin on same NCLR.
- When providing RRS may be deployed manually by ERCOT instruction or automatically by UFR.
- Will also be able to participate in Non-Spin at the end of May 2022 with manual ERCOT deployment.
- NCLR providing RRS or Non-Spin are required to consume at least the amount of RRS or Non-Spin provided unless deployed as Ancillary Service.

[5] Large Flexible Load Resource Participation in the ERCOT Region, ERCOT presentation to Large Flexible Load Task Force, April 26th, 2022, slide 6.

8. Conclusion

8.1 ERCOT Market has long history of Load Participation

Large industrial and commercial loads have participated in the ERCOT market at least since retail deregulation was implemented in 2001. Loads have participated in the ERCOT market by reducing consumption during Real-Time Market high price intervals. Load Serving Entities serving such load can pass through Four Coincident Peak (4CP) charge used to recover transmission cost of service and RTM Settlement Point Prices for corresponding Load Zone. Loads then voluntarily (i.e., without being instructed by ERCOT) respond to avoid 4CP and high RTM SPPs thereby managing their exposure to high prices.

ERCOT market is also seeing a huge influx of cryptocurrency mining datacenter load – a sizable portion of which are co-locating with renewable generators to take advantage of curtailed and low-priced energy. ERCOT market rules and policies are evolving to accommodate this rapid increase in large loads interconnection to the ERCOT grid.

8.2 Mr. McCamant's Report has Erroneous Assertion

Mr. McCamant provides a high-level description of the workings of the ERCOT market in his report. However, he makes an erroneous assertion in page 16 of his report due to ignoring the value derived by the load from consuming that energy. Ignoring this "breakeven" analysis to determine the RTM price at which to curtail would imply that loads would lose money whenever the load engaged in this uneconomic "arbitrage" and the RTM price was below the revenue generated by the load from consuming the energy. To the best of my knowledge, no load engages in such money-losing "arbitrage."

Mr. McCamant may not have been aware of such "breakeven" method where the value generated by the load by consuming the energy is taken into account in determining the price at which to curtail consumption, however, this "breakeven" method has been used by large loads ever since at least soon after retail deregulation in 2001. Thus, "breakeven" arbitrage is not a new concept but one that has been widely used in the ERCOT market for decades now.

8.3 Right to Supplement or Amend based on Discovery

I reserve the right to supplement or amend this Report based on discovery.

WGR	Wind-powered Generation Resource
WGRPP	Wind-powered Generation Resource Production Potential
WRUC	Weekly Reliability Unit Commitment
WSL	Wholesale Storage Load

Shams Siddiqi

EXHIBIT T

From: Stover, Chad <Chad.Stover@btlaw.com>
Sent: Wednesday, June 23, 2021 3:09 PM
To: Benjamin T. Horton; John R. Labbe
Cc: Nelson, Mark; Chelsea M. Murray; Mayo, Andrew C.; Ray Ricordati
Subject: RE: BearBox and Storms v. Lancium - Rule 26(f) discovery conference
Attachments: Bearbox v. Lancium - Draft Scheduling Order 6-22-21-c-c.docx; Judge Noreika letter.docx

External - This email is from an external email address outside the firm.

Ben,

Thanks for providing this and making these changes. In the attached version, I made a few minor changes that I hope won't be controversial. Here's a summary:

- Pushed the trial date back about a month because of a conflict on our end with August 2022. I also pushed back the dispositive motion deadline and pretrial conference a bit given that there was more time in schedule after moving the trial date.
- I changed the rebuttal expert report deadline from Feb. 18 to Feb 22, 2022
- I changed the RFAs to 50 and excluded RFA related solely to authenticity from this limit.
- I added a note about deposition locations. We expect we can agree to take depositions near where the witnesses live/work.

On the cover letter, please see the attached draft. We reserve the right to modify our portion in response to the arguments you add. I'm happy to put this on my letterhead after receiving your portion.

Thanks,
Chad

Chad S.C. Stover | Partner
Direct: (302) 300-3474 | Mobile: (302) 766-2932
Wilmington, DE

BARNES & THORNBURG LLP

From: Benjamin T. Horton <bhorton@marshallip.com>
Sent: Tuesday, June 22, 2021 5:36 PM
To: Stover, Chad <Chad.Stover@btlaw.com>; John R. Labbe <jlabbe@marshallip.com>
Cc: Nelson, Mark <mnelson@btlaw.com>; Chelsea M. Murray <cmurray@marshallip.com>; Mayo, Andrew C. <AMayo@ashbygeddes.com>; Ray Ricordati <rricordati@marshallip.com>
Subject: [EXTERNAL]RE: BearBox and Storms v. Lancium - Rule 26(f) discovery conference

Hi Chad,

I attach a revised proposed schedule, consistent with our call this morning. For clarity, we accepted all previous edits, and new changes are now in redline. As you requested, we moved up the trial date by two months to August 2022. We reconfigured a few of the intermediate deadlines, but maintained the spacing that we all agree is appropriate. As you suggested, we removed the claim construction dates, leaving only the August 27, 2021 deadline, a promise to inform the

Court of the parties' decision and, if Markman is necessary, that the parties will propose a claim construction schedule for the Court's consideration.

We left the RFA limit at 25, but we will agree that the limit does not apply to authentication RFAs. If that doesn't address your request to increase the limit to 50 RFAs, let me know and we can work something out.

This should address all issues with this proposal. We look forward to your confirmation that Defendants will raise bifurcation with Judge Noreika by cover letter and, if so, that we will receive your portion of that letter the first half of tomorrow.

Best regards,
Ben

From: Stover, Chad <Chad.Stover@btlaw.com>
Sent: Monday, June 21, 2021 5:02 PM
To: John R. Labbe <jlabbe@marshallip.com>; Benjamin T. Horton <bhorton@marshallip.com>
Cc: Nelson, Mark <mnelson@btlaw.com>; Chelsea M. Murray <cmurray@marshallip.com>; Mayo, Andrew C. <AMayo@ashbygeddes.com>; Ray Ricordati <rricordati@marshallip.com>
Subject: RE: BearBox and Storms v. Lancium - Rule 26(f) discovery conference

External - This email is from an external email address outside the firm.

John,

Today got away from me. Noon ET tomorrow works. I'll look forward to speaking with you then.

Thanks,
Chad

Chad S.C. Stover | Partner
Direct: (302) 300-3474 | Mobile: (302) 766-2932
Wilmington, DE

BARNES & THORNBURG LLP

From: John R. Labbe <jlabbe@marshallip.com>
Sent: Monday, June 21, 2021 3:04 PM
To: Stover, Chad <Chad.Stover@btlaw.com>; Benjamin T. Horton <bhorton@marshallip.com>
Cc: Nelson, Mark <mnelson@btlaw.com>; Chelsea M. Murray <cmurray@marshallip.com>; Mayo, Andrew C. <AMayo@ashbygeddes.com>; Ray Ricordati <rricordati@marshallip.com>
Subject: [EXTERNAL]Re: BearBox and Storms v. Lancium - Rule 26(f) discovery conference

Chad,

Ben isn't available this afternoon, but I could talk at either 4:30 or 5:00 ET. We're also available at noon ET tomorrow and other times tomorrow, as well, if necessary.

Please let us know when you would prefer to talk.

John

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX LLC and AUSTIN STORMS, Plaintiffs, v. LANCIUM LLC, MICHAEL T. MCNAMARA, and RAYMOND E. CLINE, JR., Defendants.	)	C.A. No. 21-534-MN

SCHEDULING ORDER [NON-PATENT; JURY TRIAL]

This day of _____, the Court having conducted an initial Rule 16(b) scheduling conference pursuant to Local Rule 16.1(b), and the parties having determined after discussion that the matter cannot be resolved at this juncture by settlement, voluntary mediation, or binding arbitration;

IT IS ORDERED that:

1. Rule 26(a)(1) Initial Disclosures and E-Discovery Default Standard.

Unless otherwise agreed to by the parties, the parties shall make their initial disclosures pursuant to Federal Rule of Civil Procedure 26(a)(1) within five (5) days of the date the Court entered this Order. If they have not already done so, the parties are to review the Court's Default Standard for Discovery, Including Discovery of Electronically Stored Information ("ESI"), which is posted at http://www.ded.uscourts.gov (*see* Other Resources, Default Standard for Discovery) and is incorporated herein by reference.

2. Joinder of Other Parties and Amendment of Pleadings. All motions to join other parties, and to amend or supplement the pleadings, shall be filed on or before **November 1, 2021**.

Unless otherwise ordered by the Court, any motion to join a party or motion to amend the pleadings shall be made pursuant to the procedures set forth in Paragraphs 7(g) and 8.

3. Application to Court for Protective Order. Should counsel find it will be necessary to apply to the Court for a protective order specifying terms and conditions for the disclosure of confidential information, counsel should confer and attempt to reach an agreement on a proposed form of order and submit it to the Court within ten (10) days from the date the Court enters this Order. Should counsel be unable to reach an agreement on a proposed form of order, counsel must follow the provisions of Paragraph 7(g) below.

Any proposed protective order must include the following paragraph:

> Other Proceedings. By entering this order and limiting the disclosure of information in this case, the Court does not intend to preclude another court from finding that information may be relevant and subject to disclosure in another case. Any person or party subject to this order who becomes subject to a motion to disclose another party's information designated "confidential" [the parties should list any other level of designation, such as "highly confidential," which may be provided for in the protective order] pursuant to this order shall promptly notify that party of the motion so that the party may have an opportunity to appear and be heard on whether that information should be disclosed.

4. Papers Filed Under Seal. In accordance with section G of the Revised Administrative Procedures Governing Filing and Service by Electronic Means, a redacted version of any sealed document shall be filed electronically within seven (7) days of the filing of the sealed document.

5. Courtesy Copies. The parties shall provide to the Court two (2) courtesy copies of all briefs and any other document filed in support of any briefs (*i.e.*, appendices, exhibits, declarations, affidavits etc.). This provision also applies to papers filed under seal. All courtesy copies shall be double-sided.

6. ADR Process. This matter is referred to a magistrate judge to explore the possibility of alternative dispute resolution.

7. Discovery. Unless otherwise ordered by the Court or agreed to by parties, the limitations on discovery set forth in the Federal Rules shall be strictly observed.

(a) Fact Discovery Cut Off. All fact discovery in this case shall be initiated so that it will be completed on or before December 14, 2021.

(b) Document Production. Document production shall be substantially complete by October 8, 2021.

(c) Requests for Admission. A maximum of fifty (50) requests for admission are permitted for each side. For clarity, this limit does not apply to requests for admission related solely to authenticity.

(d) Interrogatories.

i. A maximum of twenty-five (25) interrogatories, including contention interrogatories, are permitted for each side.

ii. The Court encourages the parties to serve and respond to contention interrogatories early in the case. In the absence of agreement among the parties, contention interrogatories, if filed, shall first be addressed by the party with the burden of proof. The adequacy of all interrogatory answers shall be judged by the level of detail each party provides (*i.e.*, the more detail a party provides, the more detail a party shall receive).

(e) Depositions.

i. Limitation on Hours for Deposition Discovery. Each side is limited to a total of thirty (30) hours of taking testimony by deposition upon oral examination.

ii. Location of Depositions. Any party or representative (officer, director, or managing agent) of a party filing a civil action in this district court must ordinarily be required, upon request, to submit to a deposition at a place designated within this district. Exceptions to this general rule may be made by order of the Court. A defendant who becomes a counterclaimant, cross-claimant, or third-party plaintiff shall be considered as having filed an action in this Court for the purpose of this provision.

(f) Disclosure of Expert Testimony.

i. Expert Reports. For the party who has the initial burden of proof on the subject matter, the initial Federal Rule of Civil Procedure 26(a)(2) disclosure of expert testimony is due on or before January 18, 2022. The supplemental disclosure to contradict or rebut evidence on the same matter identified by another party is due on or before February 18, 2022. Reply expert reports from the party with the initial burden of proof are due on or before March 8, 2022. No other expert reports will be permitted without either the consent of all parties or leave of the Court. Along with the submissions of the expert reports, the parties shall advise of the dates and times of their experts' availability for deposition.

ii. Objections to Expert Testimony. To the extent any objection to expert testimony is made pursuant to the principles announced in *Daubert v. Merrell Dow Pharmaceuticals, Inc.*, 509 U.S. 579 (1993), as incorporated in Federal Rule of Evidence 702, it shall be made by motion no later than the deadline for dispositive motions set forth herein, unless otherwise ordered by the Court.

iii. Expert Discovery Cut Off. All expert discovery in this case shall be initiated so that it will be completed on or before April 13, 2022.

(g) Discovery Matters and Disputes Relating to Protective Orders.

i. Any discovery motion filed without first complying with the following procedures will be denied without prejudice to renew pursuant to these procedures.

ii. Should counsel find, after a reasonable effort pursuant to Local Rule 7.1.1 that they are unable to resolve a discovery matter or a dispute relating to a protective order, the parties involved in the discovery matter or protective order dispute shall contact the Court's Judicial Administrator to schedule an argument.

iii. On a date to be set by separate order, generally not less than four (4) days prior to the conference, the party seeking relief shall file with the Court a letter, not to exceed three (3) pages, outlining the issues in dispute and its position on those issues. On a date to be set by separate order, but generally not less than three (3) days prior to the conference, any party opposing the application for relief may file a letter, not to exceed three (3) pages, outlining that party's reasons for its opposition.

iv. The parties shall provide to the Court two (2) courtesy copies of its discovery letter and any other document filed in support of any letter (*i.e.*, appendices, exhibits, declarations, affidavits etc.). This provision also applies to papers filed under seal. All courtesy copies shall be double-sided.

v. Should the Court find further briefing necessary upon conclusion of the conference, the Court will order it. Alternatively, the Court may choose to resolve the dispute prior to the conference and will, in that event, cancel the conference.

8. Claim Construction. The parties have determined that claim construction may be necessary in this case. By August 27, 2021, the parties shall determine whether or not claim construction is necessary and further notify the Court of their decision, and if claim construction is necessary, the parties will propose a claim construction schedule for the Court's consideration.

9. Motions to Amend / Motions to Strike.

(a) Any motion to amend (including a motion for leave to amend) a pleading or any motion to strike any pleading or other document shall be made pursuant to the discovery dispute procedure set forth in Paragraph 7(g) above.

(b) Any such motion shall attach the proposed amended pleading as well as a "redline" comparison to the prior pleading or attach the document to be stricken.

10. Case Dispositive Motions.

(a) All case dispositive motions, an opening brief, and affidavits, if any, in support of the motion shall be served and filed on or before June 13, 2022. Briefing will be presented pursuant to the Court's Local Rules. No case dispositive motion under Rule 56 may be filed more than ten (10) days before the above date without leave of the Court.

(b) Concise Statement of Facts Requirement. Any motion for summary judgment shall be accompanied by a separate concise statement, not to exceed six (6) pages, which details each material fact which the moving party contends is essential for the Court's resolution of the summary judgment motion (not the entire case) and as to which the moving party contends there is no genuine issue to be tried. Each fact shall be set forth in a separate numbered paragraph and shall be supported by specific citation(s) to the record.

Any party opposing the motion shall include with its opposing papers a response to the moving party's concise statement, not to exceed six (6) pages, which admits or disputes the facts set forth in the moving party's concise statement on a paragraph-by-paragraph basis. To the extent a fact is disputed, the basis of the dispute shall be supported by specific citation(s) to the record. Failure to respond to a fact presented in the moving party's concise statement of facts shall indicate that fact is not in dispute for purposes of summary judgment. The party opposing the

motion may also include with its opposing papers a separate concise statement, not to exceed four (4) pages, which sets forth material facts as to which the opposing party contends there is a genuine issue to be tried. Each fact asserted by the opposing party shall also be set forth in a separate numbered paragraph and shall be supported by specific citation(s) to the record.

The moving party shall include with its reply papers a response to the opposing party's concise statement of facts, not to exceed four (4) pages, on a paragraph-by-paragraph basis. Failure to respond to a fact presented in the opposing party's concise statement of facts shall indicate that fact remains in dispute for purposes of summary judgment.

11. Applications by Motion. Except as otherwise specified herein, any application to the Court shall be by written motion. Any non-dispositive motion should contain the statement required by Local Rule 7.1.1.

12. Motions *in Limine.* Motions *in limine* shall not be separately filed. All *in limine* requests and responses thereto shall be set forth in the proposed pretrial order. Each party shall be limited to three (3) *in limine* requests, unless otherwise permitted by the Court. The *in limine* request and any response shall contain the authorities relied upon; each *in limine* request may be supported by a maximum of three (3) pages of argument, may be opposed by a maximum of three (3) pages of argument, and the party making the *in limine* request may add a maximum of one (1) additional page in reply in support of its request. If more than one party is supporting or opposing an *in limine* request, such support or opposition shall be combined in a single three (3) page submission (and, if the moving party, a single one (1) page reply), unless otherwise ordered by the Court. No separate briefing shall be submitted on *in limine* requests, unless otherwise permitted by the Court.

13. Pretrial Conference. On **September ___, 2022**, the Court will hold a pretrial conference in Court with counsel beginning at 10:00 a.m. The parties shall file with the Court the joint proposed final pretrial order in compliance with Local Rule 16.3(c) and the Court's Preferences and Procedures for Civil Cases not later than seven (7) days before the pretrial conference. Unless otherwise ordered by the Court, the parties shall comply with the timeframes set forth in Local Rule 16.3(d)(1)-(3) for the preparation of the joint proposed final pretrial order. The Court will advise the parties at or before the above-scheduled pretrial conference whether an additional pretrial conference will be necessary.

The parties shall provide the Court two (2) double-sided courtesy copies of the joint proposed final pretrial order and all attachments. The proposed final pretrial order shall contain a table of contents and the paragraphs shall be numbered.

14. Jury Instructions, Voir Dire, and Special Verdict Forms. Where a case is to be tried to a jury, pursuant to Local Rules 47.1(a)(2) and 51.1 the parties should file (i) proposed voir dire, (ii) preliminary jury instructions, (iii) final jury instructions, and (iv) special verdict forms seven (7) full business days before the final pretrial conference. This submission shall be accompanied by a courtesy copy containing electronic files of these documents, in Microsoft Word format, which may be submitted by e-mail to mn_civil@ded.uscourts.gov.

15. Trial. This matter is scheduled for a four (4) day jury trial beginning at 9:30 a.m. on **September __, 2022**, with the subsequent trial days beginning at 9:00 a.m. Until the case is submitted to the jury for deliberations, the jury will be excused each day at 4:30 p.m. The trial will be timed, as counsel will be allocated a total number of hours in which to present their respective cases.

The Honorable Maryellen Noreika

United State District Judge

EXHIBIT A

EVENT	DEADLINE
Submission of Joint Scheduling Order	June 24, 2021
Rule 26(a)(1) Initial Disclosures	Within 5 days of the date of the Scheduling Order
Application to Court for Protective Order	Within 10 days of the date of the Scheduling Order
Determination of Whether Claim Construction is Necessary and Advise Court	August 27, 2021
Substantial Completion of Document Production	October 8, 2021
Fact Discovery Cut-Off	December 14, 2021
Initial Expert Reports	January 18, 2022
Rebuttal Expert Reports	February 22, 2022
Reply Expert Reports	March 8, 2022
Expert Discovery Cut-Off	April 13, 2022
Case Dispositive Motions	June 13, 2022
Pretrial Conference	September ___, 2022
Trial (4-day jury)	September ___, 2022

Judge Noreika,

I write on behalf of the parties in response to the Court's direction that the parties confer on a proposed scheduling order. The parties have fundamentally different positions because they disagree about what counts are properly in the case and whether a jury trial is needed on Plaintiffs' inventorship counts. Each side states its position below. The parties request a scheduling conference to further explain their positions and answer any questions the Court may have.

Lancium's Position

This case is a narrow inventorship dispute about whether Lancium's '433 patent was based on Lancium's own work or a single email and several texts sent to Lancium's CEO by Plaintiff Austin Storms, the contents of which are predated by Lancium's own work. (D.I. 23 at pp. 27-32.) In addition to Counts I and II for correction of inventorship, Plaintiffs also allege conversion, unjust enrichment, and negligent misrepresentation (Counts III-V). The allegations in Counts III-V show that inventorship is essential to the resolution of these counts. *See* D.I. 19 at ¶ 61 (conversion: alleging Lancium's dominion over the BearBox Technology by claiming it in the '433 patent; unjust enrichment: alleging Lancium unlawfully used property by asserting inventorship over the BearBox Technology; negligent misrepresentation: alleging Lancium recklessly incorporated the BearBox Technology into a patent application). Because inventorship is essential to the resolution of these claims, they are preempted and should be dismissed. *See, e.g, Speedfit LLC v. Woodway USA, Inc.*, 226 F. Supp. 3d 149, 160 (E.D.N.Y. 2016) (dismissing conversion count because it "turn[ed] on a determination of inventorship").

The pleadings closed yesterday when Plaintiffs filed their Answer to Counterclaims one day early. Lancium plans to file a Rule 12(c) motion for judgment on the pleadings to dismiss Plaintiffs' conversion, unjust enrichment, and negligent misrepresentation counts early next week. This motion will be based on preemption and other legal deficiencies in these claims. If granted, the only remaining counts will be for correction of inventorship, to which no right to a trial by jury attaches. *See, e.g., MCV, Inc. v. King-Seeley Thermos Co.*, 870 F.2d 1568, 1570 (Fed. Cir. 1989) ("section 256 [] explicitly authorizes judicial resolution of co-inventorship contests over issued patents"). While the Federal Circuit has held that commonality between an inventorship claim and a fraud claim required a trial by jury, *Shum v. Intel Corp.*, 499 F.3d 1272, 1279 (Fed. Cir. 2007), this is a different case without a fraud claim. And, in any event, Plaintiffs' other claims should be dismissed as preempted.[1]

Recognizing that it will take time to brief Lancium's motion and for the Court to decide it, Lancium proposes that the Court should allow the parties to continue discovery on the inventorship

[1] During the parties' meet and confer, Plaintiffs took the position that a jury trial is needed on all counts and a full schedule through trial should be entered now. Despite the fact that inventorship is an equitable issue, Plaintiffs contended that they have a right to a jury trial on inventorship because of factual issues in common with their conversion, unjust enrichment, and negligent misrepresentation counts. This factual commonality, however, only reinforces that Plaintiffs' conversion, unjust enrichment, and negligent misrepresentation claims are preempted by federal law and should be dismissed.

issue that has already begun and convene a scheduling conference after the Court's decision on Lancium's motion. Lancium believes this is the most efficient path forward.

While Lancium believes considering and entering a scheduling order after a decision on its motion is the most efficient path forward, the parties met and conferred and jointly submit the proposed schedule attached to this letter. If the Court decides to issue a full scheduling order now, Lancium agrees to the attached scheduling proposal while reserving the right to seek to amend that schedule depending on the outcome of its motion. For instance, if inventorship is the only surviving claim, Counts I and II could and should be tried to the bench on an abbreviated schedule.

EXHIBIT U

REDACTED IN ITS ENTIRETY

EXHIBIT V

UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX LLC and AUSTIN STORMS,	)	
Plaintiffs,	)	Civil Action No.
vs.	)	21-534-MN-CJB
LANCIUM LLC, MICHAEL T. McNAMARA, and RAYMOND E. CLINE, JR.,	)	
Defendants.	)	

VIDEO DEPOSITION OF VITOR de MIRANDA HENRIQUE

APPEARING REMOTELY

February 21, 2022

8:57 a.m. CST

REPORTED BY: WENDY A. KILLEN, CSR, RPR

LICENSE NO.: 084-003772

APPEARING REMOTELY

REMOTE APPEARANCES:

FOR THE PLAINTIFFS:

MARSHALL GERSTEIN BORUN LLP, by
RAY RICORDATI, ESQ.
233 South Wacker Drive, Suite 6300
Chicago, Illinois 60606-6357
(312) 474-6300
rricordati@marshallip.com

FOR THE DEFENDANTS:

BARNES & THORNBURG LLP, by
CHAD S.C. STOVER, ESQ.
1000 North West Street, Suite 1500
Wilmington, Delaware 19801
(302) 300-3474
chad.stover@btlaw.com

ALSO REMOTELY PRESENT:

JUSTIN HENRICKSEN, Videographer

* * * * *

INDEX TO EXAMINATION

WITNESS: VITOR de MIRANDA HENRIQUE

INDEX TO EXHIBITS

INDEX TO EXHIBITS (continued)

* * * * *

THE VIDEOGRAPHER: Good morning. We are going on the record. The time is 8:57 a.m. on February 21, 2022.

This is Media Unit 1 of the video-recorded deposition of Vitor Henrique taken by counsel for the Plaintiff in the matter of BearBox LLC, et al., versus Lancium LLC, et al., filed in the State District Court for the District of Delaware, Case Number 21-534-MN-CJB.

This deposition is being held remote Zoom videoconference.

My name is Justin Henricksen from the firm Veritext. I am the videographer. The court reporter is Wendy Killen from the firm Veritext. I'm not related to any party in this action, nor am I financially interested in the outcome.

Counsel and all present in the room, everyone attending remotely, will now state their appearance and affiliation for the record. If there are any objections to the proceedings, please state them at the time of your appearance beginning with the taking attorney first.

MR. RICORDATI: Ray Ricordati, Marshall Gerstein, representing Plaintiffs, Austin Storms and BearBox, LLC.

Page 62

avoid a frequency response that dips below a threshold?
A. That, you would have to ask for an electrical engineer because I have no idea.
Q. Okay. Fair enough.
So then for the frequency response portion of the CLE -- I'm sorry -- ELR requirement, ERCOT provides a math equation that you implemented in code; is that correct?
A. Yes.
Q. Did anyone else help out with the code of that portion of the ELR requirements?
A. Yes, probably.
Q. Who would that have been?
A. High probability that Ray Cline helped me.
Q. Did Ray Cline help you with code often?
A. No. He only provided the requirements. He --
Q. Okay. Does Ray Cline ever provide code?
A. Sorry. I was talking, too.
Q. I'm sorry. So does Ray Cline ever help you write code?
A. No.
Q. What does Ray Cline provide to you?

Page 63

A. Requirements and ideas.
Q. Then you take those requirements and turn them into functional code?
A. That's correct.
Q. Okay. Have you ever heard of the term hash and cash?
A. No.
Q. I'd like to talk about the Smart Response code for a minute.
Who besides you contributed actual source code for the Smart Response system?
A. It depends on the timestamp.
Q. Okay. So let's start at the beginning. Who was the --
A. Just me.
Q. Okay. So when did you first get help with the code portion of the Smart Response system?
A. We hired one friend of mine to join the company. I can't recall that year.
Q. And what was that person's name?
A. Danilo Martins.
Q. Did anyone else help you with the code?
A. Yes. And we hired two more people after that until today. One is Eri Elias. I believe he was hired last year. And we just hired another

Page 64

software developer, Hermano Ferreira.
Q. Okay. Besides internal Lancium employees or consultants -- I'm not exactly sure the relationship with Danilo -- but besides the people you have already mentioned, did anyone else help you with the code of the Lancium Smart Response?
A. No.
Q. Okay. So no third parties contributed any code to the Lancium Smart Response software?
A. Smart Response, no.
Q. Okay. Are third parties providing code for other feature -- other software provided by Lancium?
A. Sorry. Could you repeat that question, please?
Q. Sure.
Has other software that Lancium provides been developed by or in collaboration with third parties?
A. We already established Tier44, right?
Q. Uh-huh.
A. So that was one.
And to the best of my knowledge, that's the only one.
MR. RICORDATI: All right. I am

Page 65

introducing another exhibit, Exhibit 61.
(Whereupon, Deposition Exhibit 61 was marked for identification.)
BY MR. RICORDATI:
Q. Do you recognize this document?
A. No.
Q. Okay. You can see near the top, under the cc field, where you were included?
A. Yes.
Q. Okay. Do you see in the middle of the page where it says, You are all set up and have been offered into the LR program starting tomorrow?
A. Yes.
Q. Okay. Does that refresh your recollection about this document or Lancium's participation in the LR program?
A. No.
Q. All right. Are you aware of Lancium's participation in an LR program?
A. Yes.
Q. Okay. What is that LR program, from reviewing this document?
A. It's a load response ancillary services, as we -- I mentioned before.

EXHIBIT W

--- ATTORNEYS' EYES ONLY ---

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX, LLC and AUSTIN STORMS,	)	
	)	
Plaintiffs,	)	
	)	
vs.	)	C.A. No. 21-534-MN-CJB
	)	
LANCIUM LLC, MICHAEL T. McNAMARA, and RAYMOND E. CLINE, JR.,	)	
	)	
	)	
Defendants.	)	

The Zoom videotaped deposition of SHAMS SIDDIQI, called for examination by the Plaintiffs, taken pursuant to the Federal Rules of Civil Procedure of the United States District Courts pertaining to the taking of depositions, taken before KELLY A. BRICHETTO, CSR No. 84-3252, Certified Shorthand Reporter of the State of Illinois, reported remotely from Chicago, Illinois, on the 31st day of May, 2022, at 9:00 a.m. Central Standard Time.

Page 2

REMOTE APPEARANCES:

On behalf of the Plaintiffs:

MARSHALL, GERSTEIN & BORUN, LLP, by
MR. BENJAMIN HORTON
Willis Tower
233 South Wacker Drive
Suite 6300
Chicago, Illinois 60606
(312) 474-9579
bhorton@marshallip.com

On behalf of the Defendants:
BARNES & THORNBURG, LLC, by
MR. MARK C. NELSON
2121 North Pearl Street
Suite 700
Dallas, Texas 75201
(214) 258-4140
mnelson@btlaw.com

- - - - - - -

ALSO PRESENT:
Mr. Peter Hudson, Videographer

Page 3

TRANSCRIPT INDEX

Page 4

INDEX OF EXHIBITS

NUMBER	DESCRIPTION	IDENTIFIED
Exhibit 88	Dr. Siddiqi's report	18
Exhibit 89	ERCOT presentation	47
Exhibit 90	Slides	49
Exhibit 91	Slides	54

Page 5

THE VIDEOGRAPHER: We are on the record at 9:07 a.m. Central daylight time on May 31, 2022 beginning the remote videorecorded deposition of Shams Siddiqi taken in the matter of BearBox, LLC, et al. versus Lancium, LLC, et al. in the United States District Court for the District of Delaware, Civil Action Number 21-534-MN-CJV.

My name is Peter Hudson, the videographer, and the court reporter is Kelly Kilcoyne, both representing Veritext Midwest.

Counsel, will you please state your appearances.

MR. HORTON: My name is Benjamin Horton, counsel for the Plaintiffs.

MR. NELSON: Mark Nelson at Barnes & Thornburg, counsel for Defendants.

THE VIDEOGRAPHER: Thank you. The court reporter will swear the witness, and you may then proceed.

Page 22

creates the break-even price, and if the real-time price goes above that break-even price, they would curtail their load.
Q. And your statement there that: "The break-even method has been used by large loads ever since at least soon after retail deregulation in 2001," are you aware of large loads using a break-even method to conduct energy arbitrage?
A. Yes, I am.
Q. And what kind of loads are you aware of that have used a break-even method to arbitrage energy values starting in 2001?
A. These are like large petrochemical industries.
Q. What are the loads for those petrochemical industries doing?
A. So they, you know, they curtail whenever the price in the real-time market is high. They curtail during 4CP intervals to avoid 4CP charges, so they essentially take into account their benefit from consuming that electricity compared to the price of electricity in the market.
Q. When you say "curtail," do you mean self-curtail?

Page 23

A. Yes, self-curtail.
Q. Do you consider that arbitrage?
A. Yes.
Q. And why is it that you consider self-curtailment arbitrage?
A. Because they're taking advantage of energy that produced ahead of -- they bought ahead of time, and then they're now looking into whether it makes sense for them to, you know, monetize on the -- on the energy, you know, profits or keep producing their goods, so that sends their arbitraging, the energy market, if the price of the energy market -- to get paid back in the energy market is higher than the value of the product.
Q. Other than the petrochemical industry are you aware of any other large loads that were conducting break-even method arbitrage beginning in 2001?
A. It was -- I mean I've consulted for those petrochemical industries but other than that. You know, ERCOT has a report that tells us how many megawatts, and they claim it's large industrial, large commercial, but I'm not exactly aware of which industries they're in.
Q. I'm asking you personally, Dr. Siddiqi, which large loads other than these petrochemical industries are you aware of that have used a break-even method to

Page 24

arbitrage energy values.
A. I don't think I've personally, you know, consulted for any other large load that would qualify as this.
Q. Okay. I'm not asking whether you've consulted for any, Dr. Siddiqi. I'm asking if you're aware of any others.
MR. NELSON: Objection to form.
BY THE WITNESS:
A. As I said, you know, I'm aware through the ERCOT reports, but that doesn't specify which industry they are part of.
BY MR. HORTON:
Q. So is the answer no, you're not personally aware of any other types of large loads that are using a break-even method to arbitrage energy values?
MR. NELSON: Objection to form.
BY THE WITNESS:
A. I'd have to think about that. I haven't thought about that.
BY MR. HORTON:
Q. Well, now's the time to answer the question, Dr. Siddiqi.
A. Right. I can just say I can't recall at this

Page 25

moment.
Q. Okay. Dr. Siddiqi, in your report you discuss at various points the relationship between a resource entity and a qualified scheduling entity; right?
A. Yes.
Q. And what is that relationship generally?
A. So the qualified scheduling entity is the, you know, communication and financial settlement interface with ERCOT for all market participants, so essentially the LSE, you know, is represented by the QSE and that's how the LSE, you know, is settled with ERCOT, how they interact with ERCOT through the QSE.
Q. Would that be true for loads as well, loads with certain ERCOT designations like load resource or controllable load resource?
A. No. The load resources would interact through the QSE with ERCOT.
Q. And the controllable load resources, would they interact with ERCOT through a QSE as well?
A. Yes, they would.
Q. And so does that require that the QSE and the controllable load resource communicate with each other?
A. Yes, it does.
Q. And how are those communications done?

Page 34

producing the goods that they produce. So from that they determine okay, at what price does it make sense I mean not to produce that and actually to -- to not consume and curtail my load.

Q. Yes, but I'm asking you specifically, Dr. Siddiqi, are you aware of how other industries which I think you testified the only one that you're aware of as you sit here today is petrochemical, how it calculates a break-even value for its load.

A. I don't believe I actually helped them calculate the break-even. I just provided the -- basically consulted with them on the concept, so I wouldn't know exactly how they calculate it.

Q. Do you know generally how they calculate it?

A. It's as I described. You know, they look at the value they get out of consuming that energy.

Q. In the last sentence of Section 6.2, Dr. Siddiqi, you write: "CLR avoiding 4CP intervals can also submit RTM energy bid at the CLR's break-even price in this SCED and thus be dispatched by SCED to maximize its profit as described in Section 3.1 of this report." Do you see that?

A. Yes, I do.

Q. Okay. Is this describing energy arbitrage

Page 35

for a CLR?

A. Yes. I mean you have to read the paragraphs above which say that, you know, they've bought the energy either in the day-ahead or through bilateral means.

Q. So even though that sentence is similar to this sentence from Section 6.1, it's different because you need the context preceding that sentence; is that fair?

A. That is correct.

Q. And it's your testimony or your opinion, Dr. Siddiqi, that there's nothing new about this type of break-even energy arbitrage method; is that right?

A. That is correct.

Q. It's been around a long time?

A. That's correct.

Q. Have you had any interviews or discussions with Mr. Michael McNamara?

A. No, I have not.

Q. Have you had any discussions with Dr. Raymond Cline?

A. No, I have not.

Q. Have you been shown any documents by Lancium or Lancium's counsel that discuss Lancium's own energy arbitrage methods for its cryptocurrency mining centers?

Page 36

A. I think I've seen those kind of documents, but I can't recall the exact, you know, exact content of those.

Q. What's the general content, if you can recall?

A. I remember seeing e-mail exchanges I believe that talked about some of that, but I can't remember, you know, what exactly their method they proposed or -- yeah, I'd have to look at those documents again.

Q. But, in any event, nothing about those documents changes your opinion that break-even energy arbitrage methods have been around in ERCOT since 2001; right?

A. That is correct.

Q. Dr. Siddiqi, sticking with that last sentence in Section 6.2 of your report that talks about a CLC submitting RTM energy bids into SCED and given the context of that section performing energy arbitrage based on a break-even method, are you aware of any cryptocurrency data centers that are operating in that way?

A. I'm aware from the ERCOT presentation that CL -- there are CLR cryptomining load that have qualified as CLRs, but I'm not sure how they're calculating the

Page 37

break-even price or what they're offering into the real-time energy bid as energy -- real-time energy bids.

Q. So as you sit here today, you don't know one way or another whether there are any cryptocurrency mining loads that are performing a break-even method energy arbitrage; is that fair?

A. As a CLR; right?

Q. Correct.

A. Yeah, you could say that.

Q. Is the ERCOT presentation that you're referring to cited in your report?

A. Yes, I believe it is.

Q. Is it in -- on Page 28 of your report, footnote 5?

A. Yes.

Q. And that ERCOT presentation is dated April 26, 2022; is that right?

A. That is correct.

Q. Would that have been the first time you saw that report from ERCOT?

A. Yes, it is.

Q. Prior to seeing that report from ERCOT on April 26, 2022 were you aware of any cryptocurrency mining loads operating within ERCOT as CLRs?

Page 38

A. I was aware that there was cryptomining load, but I wasn't aware that they were qualified for CLRs.
Q. Are you aware now, Dr. Siddiqi, as you sit here today having been informed through that April 2022 ERCOT report of cryptocurrency mining loads operating within ERCOT who control those cryptocurrency mining loads?
A. I was -- yeah, I've been -- I found out that at least one of those entities is controlling those loads.
Q. And which entity's controlling those loads?
A. I believe it's Lancium.
Q. Dr. Siddiqi, if I could turn your attention to Section 6.4 of your report which begins on Page 26.
A. Yes.
Q. That section is titled: "Load Participation in ERCOT Ancillary Service Market." Do you see that?
A. Yes, I do.
Q. And if you scroll down to Page 27 of your report still under the Section 6.4 heading, do you see that you have a list of bullet points and the sentence preceding those bullet points says: "All load resources must meet the following qualifications"?
A. Yes, I see that.

Page 39

Q. And you provided a list of qualifications; right?
A. That is correct.
Q. And are you aware of any technical challenges associated with the qualifications required of the load resources you've listed here?
A. You have to have the required, you know, metering, telemetry and the ability to respond to ERCOT instructions. Those have been the biggest challenges loads face in being qualified.
Q. And are these things that the QSE will typically handle for the load?
MR. NELSON: Objection to form.
BY THE WITNESS:
A. The QSEs can assist the loads in meeting some of those requirements, yes.
BY MR. HORTON:
Q. Are these the type of qualifications that have listed for a load resource within ERCOT for a number of years?
A. Yes, they have existed for some time now.
Q. How many years?
A. Non-controllable load has been here for I think since -- at least since 2001. Controllable load, I

Page 40

can't remember exactly when we introduced the concept of controllable load, but ever since that concept has been introduced these requirements have been there.
Q. So for a load resource these qualifications have been around since at least 2001; is that right?
MR. NELSON: Objection to form.
BY THE WITNESS:
A. Well, the non-controllable load, it has been around, yes, since then. Controllable load, it's since controllable load was first introduced.
BY MR. HORTON:
Q. I believe the controllable load designation was first introduced in 2004. Does that sound right to you?
A. That could be right. I haven't looked at that information.
Q. And the list of qualifications for a CLR which you present on Page 27 under Section 6.5, do you see that?
A. Yes, I do.
Q. And would those qualifications have been present then since 2004 when the CLR designation was introduced?
MR. NELSON: Objection to form.

Page 41

BY THE WITNESS:
A. I don't know if they would be exactly the same, but something similar would have existed, yes.
BY MR. HORTON:
Q. And a load in order to meet the qualifications for CLR under Section 6.5 would need a sophisticated control system. Would you agree with that?
A. Yes. It would need a control system, yes.
Q. Would it need a sophisticated control system?
A. That's sort of a subjective term, but they need a control system that is capable of performing these functions.
Q. And a CLR to meet these qualifications even as of 2004 when that designation was introduced would need a control system like that; correct?
A. Yes. Whatever the qualifications were at that time they would have to meet those qualifications.
Q. Whatever the qualifications were at that time, they were similar to the qualifications you've listed here; is that right?
A. To the best of my recollection, I would say it would be quite similar.
Q. If you could flip down to Page 28, Dr. Siddiqi, of your report, Plaintiffs' Exhibit 88.

Page 54

patent would have any impact on proration within ERCOT; right?
MR. NELSON: Objection to form. I didn't hear your question, Ben. Could I get it back?
MR. HORTON: Kelly, could you read that back, please.
(Requested portion of the record read.)
MR. NELSON: Objection to form.
BY THE WITNESS:
A. Yes, I have no opinion on that.
BY MR. HORTON:
Q. Dr. Siddiqi, do you see a powerpoint presentation marked as Exhibit 91?
A. Yes, I do.
Q. Do you recognize this document?
A. I believe so.
Q. Is this a document you would have relied upon in forming your opinions in your report and cited this document? We're going to flip through it.
A. Okay. Okay.
Q. Do you see slide three of this presentation?
A. Yes, I do.
Q. In the first bullet point ERCOT writes that:

Page 55

"Resources fall into two basic categories: Generation resources and load resources." Do you see that?
A. Yes, I do.
Q. Do you agree with that?
A. Yes.
Q. And then underneath load resources ERCOT describes two categories of load resources: Controllable load resources and non-controllable load resources. Do you see that?
A. Yes, I do.
Q. Do you agree with that?
A. Yes, I do.
Q. And staying with slide three, the third major bullet point down says: "Qualified scheduling entities," QSE. Do you see that?
A. Yes, I do.
Q. Here do you understand that ERCOT is listing five bullet points of the basic operation of the QSE?
A. Yes.
Q. Do you agree with those five bullet points?
A. Yes.
MR. HORTON: Okay. Could we take another quick five-minute break? I might be done, Mark.
MR. NELSON: Correct.

Page 56

THE VIDEOGRAPHER: We're off the record at 10:52 a.m.
(WHEREUPON, a break was taken.)
We are back on the record at 10:56 a.m.
BY MR. HORTON:
Q. Dr. Siddiqi, you testified that using a break-even value method of energy arbitrage is nothing new within the ERCOT market; is that right?
A. Yes, that is correct.
Q. In your opinion, would it be possible to have a specific way of determining a break-even value for a load in order to conduct energy arbitrage that would be new and valuable?
MR. NELSON: Objection to form.
BY THE WITNESS:
A. I think the concept is the same. For each industry it's a little different how you calculate the break-even, but the concept is exactly the same across industries.
BY MR. HORTON:
Q. The question to you, Dr. Siddiqi, is in your opinion would it be possible to develop a new and valuable way of determining a break-even value for energy

Page 57

arbitrage within a particular industry.
MR. NELSON: Objection to form.
BY THE WITNESS:
A. Would you expand on that? What will be new or how would it be different in other industries?
BY MR. HORTON:
Q. Just asking you if it's possible in your opinion.
Would you like the court reporter to read the question back?
THE WITNESS: Yes, please.
MR. HORTON: Kelly, can you read that, please.
(Requested portion of the record read.)
BY THE WITNESS:
A. You know, sir, as I said, the concept of break-even is standard but each industry is different, so within that industry you could possibly develop a way to -- to determine that break-even, but I would assume that all participants in that industry would do the same.
BY MR. HORTON:
Q. Okay. Just to be clear, Dr. Siddiqi, in your opinion, it would be possible to come up with a new and valuable way of determining break-even value to conduct

energy arbitrage; right?
MR. NELSON: Objection to form.
BY THE WITNESS:
A. Well, that -- if it's a new industry and I guess if no one's done it before, yeah, you could come up with a new way of doing it.
MR. HORTON: Thanks. No further questions.
MR. NELSON: All right. Before I forget, I don't know if there was anything confidential discussed in this, but let's mark the transcript Attorneys Eyes' Only for now just in case there was. And why don't we take about a five-minute break and let me see if I have any follow-up?
THE VIDEOGRAPHER: We're off the record at 10:59 a.m.
(WHEREUPON, a break was taken.)
We are back on the record at 11:01 a.m.
MR. NELSON: Dr. Siddiqi, I have no questions at this time, and I thank you for your time.
THE WITNESS: Okay. Thank you.
THE VIDEOGRAPHER: This then concludes the deposition of Shams Siddiqi on May 31, 2022. The time is 11:02 a.m. Central, and we are off the record.

SIGNATURE:
It was agreed by and between counsel and the parties that the Deponent will read and sign the transcript of said deposition.

STATE OF ILLINOIS)
) SS:
COUNTY OF C O O K)

I, KELLY A. BRICHETTO, a Certified Shorthand Reporter of said state, do hereby certify that the within named witness, SHAMS SIDDIQI, was by me first duly sworn to testify the truth, the whole truth and nothing but the truth in the cause aforesaid; that the testimony then given by the above-referenced witness was by me reduced to stenotype in the presence of said witness; afterwards transcribed, and that the foregoing is a true and correct transcription of the testimony so given by the above-referenced witness.

I do further certify that this deposition was taken at the time and place in the foregoing caption specified and was completed without adjournment.

I do further certify that I am not a relative, counsel or attorney for either party or otherwise interested in the event of this action.

IN WITNESS WHEREOF, I do hereunto set my hand this 13th day of June, 2022.

Kelly Brichetto

KELLY A. BRICHETTO
CSR License No. 84-3252

EXHIBIT X

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

BEARBOX LLC, et al.,)
)
Plaintiffs,) C.A. No. 21-534-MN-CJB
)
v.)
)
LANCIUM LLC, et al.,)
)
Defendants.)

Wednesday, September 8, 2021
11:00 a.m.

844 King Street
Wilmington, Delaware

BEFORE: THE HONORABLE CHRISTOPHER J. BURKE
United States District Court Judge

APPEARANCES:

ASHBY & GEDDES
BY: ANDREW MAYO, ESQ.

-and-

MARSHALL, GERSTEIN & BORUN
BY: JOHN LABBE, ESQ.
BY: BENJAMIN HORTON, ESQ.

Counsel for the Plaintiff

APPEARANCES CONTINUED:

BARNES & THORNBURG
BY: CHAD STOVER, ESQ.
BY: ADAM KAUFMANN, ESQ.

Counsel for the Defendant

THE COURT: Good morning, everyone, this is Judge Burke. Before we begin, let me just say a few things for the record. And the first is that we're here this morning by way of a teleconference to resolve a protective order dispute in the matter of BearBox LLC, et al., versus Lancium LLC, et al., civil action number 21-534-MN here in our court. Before we go further, let's have counsel for each side identify themselves for the record. We'll start first with counsel for the plaintiff's side and we'll begin there with Delaware counsel.

MR. MAYO: Good morning, Your Honor. This is Andrew Mayo from Ashby & Geddes on behalf of the plaintiffs. I am joined on the telephone by my co-counsel from Marshall, Gerstein & Borun. You have John Labbe and Benjamin Horton on the line and Mr. Labbe will be handling the argument for the plaintiff's today.

THE COURT: Okay. And we'll do the same with defendant's side and again we'll with Delaware counsel.

MR. STOVER: Sure. Good morning,

Your Honor. This is Chad Stover from Barnes & Thornburg for defendants Lancium, Mr. Cline and Mr. McNamara. And with me is Adam Kaufmann, my partner from Texas.

THE COURT: All right. Thank you. And good morning to everyone. All right. Counsel, I've read the parties' joint letter and I'll consider defendant's side to be the movant because they're the ones who are seeking the more restrictive provision in the protective order and so they'll have the burden. And so let me turn first to defendant's counsel who is going to be addressing this issue and is that Mr. Kaufmann?

MR. STOVER: It's actually going to be me, Your Honor, Mr. Stover.

THE COURT: Okay. Mr. Stover, so with regard to -- let me just ask a couple of quick questions and then we'll -- I'll certainly give you the chance to add anything that you wish to add to this dispute. I guess first is, just kind of a standard here, my sense is that -- all of the parties didn't mention it, that it is the case under the law that because

And in that situation, the grid operator may want to do something called curtailment. And curtailment is when the power generator reduces output from what it would otherwise produce in order to try to better match the load. And to achieve curtailment the grid operator might decrease the market price, for instance, being paid to generate power to reduce the incentive to generate power. So that's one way to do it. And if that doesn't work, there are other ways. In fact, it's possible for the grid operator to instruct the generator to stop producing power in order to avoid a situation where the grid could have an unstable situation. And so that's the problem with curtailment is that it can result and usually does result in inefficient power generation or sometimes even wasted power, because there's just not an ability to put that power into the grid. And so everyone wants to avoid a curtailment situation. So that's the grid side.

So on the bitcoin mining side, the bitcoin, sometimes referred to as crypto-currency, bitcoin mining is a process by

which new bitcoins are entered into circulation and it's also how the block chain ledger is maintained. It's performed using computers and they solve extremely complex computational math problems. And so miners, they have their own computer or computers are rewarded for their work in helping to maintain the block chain ledger with crypto tokens, so basically they're paid in the bitcoin crypto-currency and bitcoins for their work. Running the computers that do these complex math problems takes a lot -- a ton of electrical power and annually, for instance, bitcoin mining uses as much electricity as the entire country of Finland, an enormous amount of power used by bitcoin mining. The mining computers are flexible in that they can be turned on and off or they can be ramped up or down varying the amount of power they consume. You can see how the bitcoin mining can be advantageous to the grid operators.

So the inventions that are at issue, and this is an inventorship case, the inventions in this '433 Patent that's at issue relate to the use of these flexible data centers

like bitcoin mining computers to use excess or low cost power from electricity generators which help keeps the grid stable and allow the miners to be profitable. So that's the technological background of the inventions that are at issue here in this '433 Patent.

Lancium, my client, is a market leader in this industry. It's a fast moving industry and potentially a very lucrative industry. Lancium has many patents and quite a few pending patent applications protecting its position that it's the leading innovator in the field. Mr. Storms and his company, BearBox, have sued my client, Lancium, the CEO, Michael McNamara and Ray Cline, who is the chief technology officer of Lancium. And Mr. Storms claims he met Mr. McNamara at a conference in 2019, that they both attended a group dinner at that conference and Mr. Storms then sent Mr. McNamara an e-mail and a few text messages. Based on that, he claims that he should be listed as an inventor or perhaps even a sole inventor of the '433 Patent by Lancium. My client believes nothing Mr. Storms told Mr.

McNamara was new. Was all old info, all things Mr. McNamara already knew and had known for quite some time. Mr. Storms claims that -- Lancium believes that Mr. Storms' claims are not only untrue but rather far-fetched, specifically the timing of stuff because Mr. Storms knew about the '433 Patent for many months, but waited to file suit until after the press release about settlement of an infringement suit that Lancium had filed involving the '433 Patent.

To Lancium this case looks like an attempted money grab or worse, actually for Lancium, a potential fishing expedition into their highly confidential information by a potential competitor.

So that's the backdrop and those are the reasons, Your Honor, why Lancium is very concerned about the disclosure of its confidential attorney eyes only information. And the restriction that we're asking for here, which is -- well, restriction is the wrong word. It's not a restriction. The acknowledgment that we're seeking, this sentence that we want to put

EXHIBIT Y

REDACTED IN ITS ENTIRETY

EXHIBIT Z

REDACTED IN ITS ENTIRETY

Exhibit AA

From: Raymond Cline <recline@lancium.com>
To: "ian.rock" <ian.rock@lancium.com>
Subject: Re: Contract started today
Date: Fri, 16 Aug 2019 11:40:57 -0500

Yep, we now make money when we shutdown.

Raymond E. Cline Jr., PhD
Chief Computing Officer

On Fri, Aug 16, 2019 at 10:58 AM <ian.rock@lancium.com> wrote:

> Ray
>
> Just FYI
>
> Ian Rock
> Director IT Operations
> LANCIUM LC
> 713 839 5246
>
> Begin forwarded message:
>
> **From:** Todd Wilson <Todd.Wilson@calpinesolutions.com>
> **Date:** August 16, 2019 at 10:31:38 AM CDT
> **To:** "Michael McNamara " <michael.mcnamara@lancium.com>, "Ian Rock (ian.rock@lancium.com)" <ian.rock@lancium.com>, "Vitor Henrique (vitor.henrique@lancium.com)" <vitor.henrique@lancium.com>
> **Subject: Contract started today**
>
> Michael, your contract pricing started today and, with high prices forecast, good day to sell power back into the market.
>
> Best Regards,
>
> Todd
>
> Todd R. Wilson, CEM
> Sales Director, Calpine Energy Solutions
>
> Cell: 919-414-7986
> E-Mail: todd.wilson@calpinesolutions.com<mailto:todd.wilson@calpinesolutions.com>
> [cid:image002.png@01D291B4.0F84FC00]
>
>

```
#!/usr/bin/env python3

import json, sqlite3, datetime, time

from datetime import datetime

from datetime import timedelta

from pathlib import Path

from bitcoin.rpc import RawProxy


folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")


def get_block_height_local():

    try:

        p = RawProxy()

        info = p.getblockchaininfo()

        block_height = (info['blocks'])


        header_hash = (info['bestblockhash'])


        header = p.getblock(header_hash)


        block_time = header['time']

        print(str(block_time))


        header_block_time = datetime.utcfromtimestamp(block_time).strftime('%Y-%m-%d %H:%M:%S')

        print(header_block_time)


        conn = sqlite3.connect(db_file)

        conn.execute("INSERT INTO block_height_data (date_time, block_height, header_block_time) "

                     "VALUES (?, ?, ?);", (str(datetime.utcnow()), block_height, header_block_time))

        conn.commit()
```

```
63      except Exception as e:
64
65              print("Error:  " + str(e))
66
67
68
69  while True:
70
71      get_block_height_local()
72
73      #print(str(datetime.datetime.now()))
74
75      time.sleep(.1)
```

```
#!/usr/bin/env python3

import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config

from pathlib import Path

from LMP_csv_import import get_real_time_LMP

from DA_LMP_import import get_day_ahead_LMP

from toggle_relay import *

from pymodbus.client.sync import ModbusTcpClient as ModbusClient

from email_alert import *

from bitcoin.rpc import RawProxy


urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)


folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")

miner_hashrate = 31000000 * 272

kW_load = 1.8 * 272

location = 'AEC'

block_reward = 12.5


def get_BTC_price():

    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')

    price_dict = price.json()

    BTC_price = price_dict['data']['amount']

    BTC_price = float(BTC_price)

    return BTC_price


def get_network_difficulty():

    try:

        p = RawProxy()

        diff = float(p.getdifficulty())


        return diff

```

```
65
66
67      except Exception as e:
68
69          print("Error:  " + str(e))
70
71
72
73  def get_block_height():
74
75      try:
76
77          p = RawProxy()
78
79          info = p.getblockchaininfo()
80
81          block_height = (info['blocks'])
82
83
84
85          return block_height
86
87
88
89      except Exception as e:
90
91              print("Error:  " + str(e))
92
93
94
95  def get_network_hashrate(diff):
96
97      network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
98
99      hashrate_dict = network_hashrate.json()
100
101     # print(hashrate_dict)
102
103     hashrate = hashrate_dict['data']['hashrate']
104
105     hashrate = float(hashrate) / 1000000
106
107     # hashrate = int(hashrate)
108
109     # print("Called network hashrate: " + str(hashrate))
110
111     return hashrate
112
113
114
115 def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
116
117     breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
118
119     #print(breakeven)
120
121     return breakeven
122
123
124
125 def pdu1_all_on():
126
127     from pymodbus.client.sync import ModbusTcpClient as ModbusClient
128
```

```
129       #mport config
130
131       client = ModbusClient(config.pdu1_ip, config.pdu1_port)
132
133       client.write_coil(0, False)
134
135       client.write_coil(1, False)
136
137       client.write_coil(2, False)
138
139       client.write_coil(3, False)
140
141       client.write_coil(4, False)
142
143       client.write_coil(5, False)
144
145       client.write_coil(6, False)
146
147       client.write_coil(7, False)
148
149       client.write_coil(8, False)
150
151       client.write_coil(9, False)
152
153       client.write_coil(10, False)
154
155       client.write_coil(11, False)
156
157       client.write_coil(12, False)
158
159       client.write_coil(13, False)
160
161       client.write_coil(14, False)
162
163       client.write_coil(15, False)
164
165       client.write_coil(16, False)
166
167       client.write_coil(17, False)
168
169       client.write_coil(18, False)
170
171       client.write_coil(19, False)
172
173       client.write_coil(20, False)
174
175       client.write_coil(21, False)
176
177       client.write_coil(22, False)
178
179       client.write_coil(23, False)
180
181
182
183       client.close()
184
185
186
187   def pdu1_all_off():
188
189       from pymodbus.client.sync import ModbusTcpClient as ModbusClient
190
191       # mport config
192
193       client = ModbusClient(config.pdu1_ip, config.pdu1_port)
```

```
194
195         client.write_coil(0, True)
196
197         client.write_coil(1, True)
198
199         client.write_coil(2, True)
200
201         client.write_coil(3, True)
202
203         client.write_coil(4, True)
204
205         client.write_coil(5, True)
206
207         client.write_coil(6, True)
208
209         client.write_coil(7, True)
210
211         client.write_coil(8, True)
212
213         client.write_coil(9, True)
214
215         client.write_coil(10, True)
216
217         client.write_coil(11, True)
218
219         client.write_coil(12, True)
220
221         client.write_coil(13, True)
222
223         client.write_coil(14, True)
224
225         client.write_coil(15, True)
226
227         client.write_coil(16, True)
228
229         client.write_coil(17, True)
230
231         client.write_coil(18, True)
232
233         client.write_coil(19, True)
234
235         client.write_coil(20, True)
236
237         client.write_coil(21, True)
238
239         client.write_coil(22, True)
240
241         client.write_coil(23, True)
242
243
244
245         client.close()
246
247
248
249     def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
250
251         try:
252
253             if(day_ahead_LMP >= breakeven):
254
255                 #pdu1_all_off()
256
257                 email_alert_RT_LMP_miners_off(real_time_LMP, breakeven)
258
```

```
259                 return
260
261             elif(real_time_LMP > breakeven):
262
263                 #pdu1_all_off()
264
265                 email_alert_RT_LMP_miners_off(real_time_LMP, breakeven)
266
267                 return
268
269             else:
270
271                 #pdu1_all_on()
272
273                 email_alert_miners_on_LMP(day_ahead_LMP, real_time_LMP, breakeven)
274
275
276         except Exception as e:
277
278             print("Error:  " + str(e))
279
280
281
282     def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
283
284         day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
285
286         day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))
287
288
289
290         real_time_LMP_rev = real_time_LMP * kW_load/(12)
291
292         real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))
293
294
295
296         mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
297
298         mining_rev = float("{0:.7f}".format(mining_rev))
299
300
301
302         return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev
303
304
305
306     def main():
307
308         try:
309
310             block_height = get_block_height()
311
312             print("Current Block Height:          " + str(block_height))
313
314
315
316             BTC_price = get_BTC_price()
317
318             BTC_price = float("{0:.2f}".format(BTC_price))
319
320             print("Current BTC Price:             " + str(BTC_price))
321
322
323
```

```
324
325         diff = get_network_difficulty()
326
327         print("Network difficulty:            " + str(diff))
328
329
330
331         hashrate = get_network_hashrate(diff)
332
333         print("Estimated Network Hashrate:    " + str(hashrate))
334
335
336
337         day_ahead_LMP = get_day_ahead_LMP(location)
338
339         day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
340
341         print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
342
343
344
345         real_time_LMP = get_real_time_LMP(location)
346
347         real_time_LMP = float("{0:.7f}".format(real_time_LMP))
348
349         print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
350
351
352
353         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
            kW_load)
354
355         breakeven = float("{0:.7f}".format(breakeven))
356
357         print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
358
359
360
361         profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
362
363
364
365         day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
            hashrate, real_time_LMP, day_ahead_LMP)
366
367
368
369         if(day_ahead_LMP >= breakeven):
370
371             realized_rev = day_ahead_LMP_rev
372
373         elif(real_time_LMP >= breakeven):
374
375             realized_rev = real_time_LMP_rev
376
377         else:
378
379             realized_rev = mining_rev
380
381
382
383         conn = sqlite3.connect(db_file)
384
385         conn.execute("INSERT INTO denis_logic_test_newgen (datetime, day_ahead_LMP,
            block_height, BTC_price, "
```

```
386
387                         "network_diff, est_network_hashrate, real_time_LMP,
                            breakeven_mining_cost, day_ahead_LMP_rev, real_time_LMP_rev,
                            mining_rev, realized_rev) "
388
389                         "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
390
391                         (str(datetime.datetime.now()), day_ahead_LMP, block_height,
                            BTC_price,
392
393                          diff, hashrate, real_time_LMP, breakeven, day_ahead_LMP_rev,
                             real_time_LMP_rev, mining_rev, realized_rev))
394
395             conn.commit()
396
397
398
399         except Exception as e:
400
401             print("Error:  " + str(e))
402
403
404
405     while True:
406
407         main()
408
409         time.sleep(300)
410
411
```

```
#!/usr/bin/env python3

import json, sqlite3, datetime, time

from datetime import datetime

from pathlib import Path

from bitcoin.rpc import RawProxy



folder = Path("C:/Users/Admin") # this is your path to the sqlite3 db

db_file = str(folder / "test_10.sqlite")  # hover mouse over db creation for URI


def get_block_height_local():

    try:

        p = RawProxy() # connect to node

        info = p.getblockchaininfo() # getblockchaininfo call

        block_height = (info['blocks']) # gets block height from getblockchaininfo



        header_hash = (info['bestblockhash']) # gets the bestblockhash from your node



        header = p.getblock(header_hash) # calls getblock and passes in the header_hash
        param from above



        block_time = header['time'] # unix timestamp of node's best block header

        block_time_utc = datetime.utcfromtimestamp(block_time).strftime('%Y-%m-%d
        %H:%M:%S') # convert unix timestamp to utc



        time_now = time.time() # call time.time() and get unix timestamp of current time

        time_now_utc = datetime.utcfromtimestamp(time_now).strftime('%Y-%m-%d
        %H:%M:%S') # convert current time unix timestamp to utc



        prop_time = (time_now - block_time) # calculate  the difference between current
        time and best block time

        prop_time_utc = datetime.utcfromtimestamp(prop_time).strftime('%H:%M:%S') #
        convert unix propagation time to utc in hours/minutes/seconds format



        conn = sqlite3.connect(db_file) # connect to sqlite3 db ---- see path and file
        under imports section in lines 7-8

        conn.execute("INSERT INTO block_height_data_test (block_height,
```

```
        header_block_time, date_time, prop_time) " # insert into table name
        'block_height_data_test', column names, values
60
61                      "VALUES (?, ?, ?, ?);", (block_height, block_time_utc,
                        time_now_utc, prop_time_utc))
62
63          conn.commit() # commit changes to db table
64
65
66
67      except Exception as e: # catch exceptions - specifically, block_height has a unique
        constraint in the created sqlite3 table
68
69              print("Error:  " + str(e))
70
71
72
73  while True:
74
75      get_block_height_local() # call the above
76
77      time.sleep(.1) # delay .1 seconds in while loop
```

```python
#!/usr/bin/env python3

import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config

from pathlib import Path

from LMP_csv_import import get_real_time_LMP

from DA_LMP_import import get_day_ahead_LMP

from toggle_relay import *

from pymodbus.client.sync import ModbusTcpClient as ModbusClient

from email_alert import *

from bitcoin.rpc import RawProxy


urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)


folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")

miner_hashrate = 14300000 * 272

kW_load = 1.3 * 272

location = 'AEC'

block_reward = 12.5


def get_BTC_price():

    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')

    price_dict = price.json()

    BTC_price = price_dict['data']['amount']

    BTC_price = float(BTC_price)

    return BTC_price


def get_network_difficulty():

    try:

        p = RawProxy()

        diff = float(p.getdifficulty())


        return diff

```

```
65
66
67      except Exception as e:
68
69          print("Error:  " + str(e))
70
71
72
73  def get_block_height():
74
75      try:
76
77          p = RawProxy()
78
79          info = p.getblockchaininfo()
80
81          block_height = (info['blocks'])
82
83
84
85          return block_height
86
87
88
89      except Exception as e:
90
91              print("Error:  " + str(e))
92
93
94
95  def get_network_hashrate(diff):
96
97      network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
98
99      hashrate_dict = network_hashrate.json()
100
101     # print(hashrate_dict)
102
103     hashrate = hashrate_dict['data']['hashrate']
104
105     hashrate = float(hashrate) / 1000000
106
107     # hashrate = int(hashrate)
108
109     # print("Called network hashrate: " + str(hashrate))
110
111     return hashrate
112
113
114
115 def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
116
117     breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
118
119     #print(breakeven)
120
121     return breakeven
122
123
124
125 def pdu1_all_on():
126
127     from pymodbus.client.sync import ModbusTcpClient as ModbusClient
128
```

```
129       #mport config
130
131       client = ModbusClient(config.pdu1_ip, config.pdu1_port)
132
133       client.write_coil(0, False)
134
135       client.write_coil(1, False)
136
137       client.write_coil(2, False)
138
139       client.write_coil(3, False)
140
141       client.write_coil(4, False)
142
143       client.write_coil(5, False)
144
145       client.write_coil(6, False)
146
147       client.write_coil(7, False)
148
149       client.write_coil(8, False)
150
151       client.write_coil(9, False)
152
153       client.write_coil(10, False)
154
155       client.write_coil(11, False)
156
157       client.write_coil(12, False)
158
159       client.write_coil(13, False)
160
161       client.write_coil(14, False)
162
163       client.write_coil(15, False)
164
165       client.write_coil(16, False)
166
167       client.write_coil(17, False)
168
169       client.write_coil(18, False)
170
171       client.write_coil(19, False)
172
173       client.write_coil(20, False)
174
175       client.write_coil(21, False)
176
177       client.write_coil(22, False)
178
179       client.write_coil(23, False)
180
181
182
183       client.close()
184
185
186
187   def pdu1_all_off():
188
189       from pymodbus.client.sync import ModbusTcpClient as ModbusClient
190
191       # mport config
192
193       client = ModbusClient(config.pdu1_ip, config.pdu1_port)
```

```
194
195         client.write_coil(0, True)
196
197         client.write_coil(1, True)
198
199         client.write_coil(2, True)
200
201         client.write_coil(3, True)
202
203         client.write_coil(4, True)
204
205         client.write_coil(5, True)
206
207         client.write_coil(6, True)
208
209         client.write_coil(7, True)
210
211         client.write_coil(8, True)
212
213         client.write_coil(9, True)
214
215         client.write_coil(10, True)
216
217         client.write_coil(11, True)
218
219         client.write_coil(12, True)
220
221         client.write_coil(13, True)
222
223         client.write_coil(14, True)
224
225         client.write_coil(15, True)
226
227         client.write_coil(16, True)
228
229         client.write_coil(17, True)
230
231         client.write_coil(18, True)
232
233         client.write_coil(19, True)
234
235         client.write_coil(20, True)
236
237         client.write_coil(21, True)
238
239         client.write_coil(22, True)
240
241         client.write_coil(23, True)
242
243
244
245         client.close()
246
247
248
249     def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
250
251         try:
252
253             if(day_ahead_LMP >= breakeven):
254
255                 pdu1_all_off()
256
257                 email_alert_RT_LMP_miners_off(real_time_LMP, breakeven)
258
```

```
            return

        elif(real_time_LMP > breakeven):

            pdu1_all_off()

            email_alert_RT_LMP_miners_off(real_time_LMP, breakeven)

            return

        else:

            pdu1_all_on()

            email_alert_miners_on_LMP(day_ahead_LMP, real_time_LMP, breakeven)


    except Exception as e:

        print("Error:  " + str(e))


def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):

    day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)

    day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))


    real_time_LMP_rev = real_time_LMP * kW_load/(12)

    real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))


    mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)

    mining_rev = float("{0:.7f}".format(mining_rev))


    return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev


def main():

    try:

        block_height = get_block_height()

        print("Current Block Height:          " + str(block_height))


        BTC_price = get_BTC_price()

        BTC_price = float("{0:.2f}".format(BTC_price))

        print("Current BTC Price:             " + str(BTC_price))
```

```
324
325         diff = get_network_difficulty()
326
327         print("Network difficulty:            " + str(diff))
328
329
330
331         hashrate = get_network_hashrate(diff)
332
333         print("Estimated Network Hashrate:    " + str(hashrate))
334
335
336
337         day_ahead_LMP = get_day_ahead_LMP(location)
338
339         day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
340
341         print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
342
343
344
345         real_time_LMP = get_real_time_LMP(location)
346
347         real_time_LMP = float("{0:.7f}".format(real_time_LMP))
348
349         print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
350
351
352
353         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
            kW_load)
354
355         breakeven = float("{0:.7f}".format(breakeven))
356
357         print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
358
359
360
361         profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
362
363
364
365         day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
            hashrate, real_time_LMP, day_ahead_LMP)
366
367
368
369         if(day_ahead_LMP >= breakeven):
370
371             realized_rev = day_ahead_LMP_rev
372
373         elif(real_time_LMP >= breakeven):
374
375             realized_rev = real_time_LMP_rev
376
377         else:
378
379             realized_rev = mining_rev
380
381
382
383         conn = sqlite3.connect(db_file)
384
385         conn.execute("INSERT INTO denis_logic_test (datetime, day_ahead_LMP,
            block_height, BTC_price, "
```

```
386
387                     "network_diff, est_network_hashrate, real_time_LMP,
                        breakeven_mining_cost, day_ahead_LMP_rev, real_time_LMP_rev,
                        mining_rev, realized_rev) "
388
389                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
390
391                     (str(datetime.datetime.now()), day_ahead_LMP, block_height,
                        BTC_price,
392
393                      diff, hashrate, real_time_LMP, breakeven, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev))
394
395         conn.commit()
396
397
398
399     except Exception as e:
400
401         print("Error:  " + str(e))
402
403
404
405 while True:
406
407     main()
408
409     time.sleep(300)
410
411
```

```python
#!/usr/bin/env python3

import smtplib, ssl, socket, json, requests, urllib3, sqlite3, os, time, datetime

from pathlib import Path

from LMP_csv_import import get_realtime_LMP

from bitcoin.rpc import RawProxy


urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)


folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")

miner_hashrate = 14500000*272

kW_load = 1.3*272

location = 'AEC'

block_reward = 12.5


def get_BTC_price():

    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')

    price_dict = price.json()

    #print(price_dict)

    BTC_price = price_dict['data']['amount']

    BTC_price = float(BTC_price)

    #print("BTC/USD price   : $" + str(BTC_price))


    return BTC_price


def get_network_difficulty_local():

    try:

        p = RawProxy()

        diff = float(p.getdifficulty())


        return diff


    except Exception as e:
```

```
66
67              print("Error:  " + str(e))
68
69
70
71  def get_network_diff_external():
72
73      try:
74
75          network_difficulty =
            requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
76
77          diff_dict = network_difficulty.json()
78
79          #print(diff_dict)
80
81          diff = diff_dict['difficulty']
82
83          #print("Difficulty: " + str(diff))
84
85
86
87          return diff
88
89
90
91      except Exception as e:
92
93              print("Error:  " + str(e))
94
95
96
97  def get_block_height_local():
98
99      try:
100
101         p = RawProxy()
102
103         info = p.getblockchaininfo()
104
105         block_height = (info['blocks'])
106
107         #print(info)
108
109
110
111         print(datetime.datetime.utcfromtimestamp(time).strftime('%Y-%m-%d %H:%M:%S'))
112
113
114
115         print(block_height)
116
117
118
119         return block_height
120
121
122
123     except Exception as e:
124
125             print("Error:  " + str(e))
126
127
128
129 def get_block_height_external():
```

```
130
131         try:
132
133             block_height
                =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
134
135             bh_dict = block_height.json()
136
137             #print(bh_dict)
138
139             block_height = bh_dict['info']['blocks']
140
141             block_height = int(block_height)
142
143             print(str(block_height))
144
145
146
147             return block_height
148
149
150
151         except Exception as e:
152
153                 print("Error:  " + str(e))
154
155
156
157     def get_network_hashrate(diff):
158
159         network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
160
161         hashrate_dict = network_hashrate.json()
162
163         #print(hashrate_dict)
164
165         hashrate = hashrate_dict['data']['hashrate']
166
167         hashrate = float(hashrate)/1000000
168
169         return hashrate
170
171
172
173     def get_profit(BTC_price, hashrate, lmp_price):
174
175         gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
176
177         print("5m gross USD profit:   $" + str(gross_profit_USD))
178
179
180
181         power_cost_USD = kW_load*lmp_price/6/1000
182
183         print("5m electricity cost:   $" + str(power_cost_USD))
184
185
186
187         net_profit_USD = gross_profit_USD - power_cost_USD
188
189         print("5m net USD profi/loss: $" + str(net_profit_USD))
190
191
192
193         net_profit_BTC = net_profit_USD/BTC_price
```

```
194
195         print("5m net BTC profit:     " + str(net_profit_BTC))
196
197
198
199         return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
200
201
202
203     def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
204
205         breakeven = ((miner_hashrate/hashrate)*(block_reward*144)*(BTC_price))/(kW_load*24)
206
207         print(breakeven)
208
209         return breakeven
210
211
212
213     def main():
214
215         print(str(datetime.datetime.now()))
216
217         block_height_local = get_block_height_local()
218
219         BTC_price = get_BTC_price()
220
221         diff_local = get_network_difficulty_local()
222
223         hashrate = get_network_hashrate(diff_local)
224
225         lmp_price = get_realtime_LMP(location)
226
227         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load)
228
229         gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff_local),
230
231                                                     lmp_price=get_realtime_LMP(location))
232
233
234
235         if(pc_USD > gp_USD):
236
237             np_USD = 0
238
239
240
241         else:
242
243             np_USD = np_USD
244
245
246
247         conn = sqlite3.connect(db_file)
248
249         conn.execute("INSERT INTO lmp_model_test (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
250
251                      "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height_local,
                         difficulty_local, breakeven) "
252
253                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
```

```
254
255                                                     (lmp_price/1000), gp_USD,
                                                        pc_USD, np_USD, np_BTC,
                                                        block_height_local, diff_local,
                                                        breakeven))
256
257         conn.commit()
258
259
260
261     while True:
262
263         main()
264
265         time.sleep(300)
```

```
import requests, csv, io, sqlite3, urllib3

import pandas as pd

from pathlib import Path



urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)



location = 'AEC'



def get_real_time_LMP(location):

    folder = Path("C:/Users/Admin")

    db_file = str(folder / "test_10.sqlite")

    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-
    SL-latestInterval.csv'



    data = requests.get(url, verify=False)



    df = pd.read_csv(io.StringIO(data.text))

    #print(df.head())



    df.to_csv('data.csv')



    conn = sqlite3.connect(db_file)



    conn.execute("DROP TABLE IF EXISTS lmp_data")

    conn.execute("CREATE TABLE lmp_data (Interval, GMTIntervalEnd, `Settlement
    Location`, Pnode, LMP, MLC, MCC, MEC);")



    with open('data.csv','r') as fin:

        dict_read = csv.DictReader(fin)

        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]



    conn.executemany("INSERT INTO lmp_data (Interval, GMTIntervalEnd, `Settlement
    Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?, ?);", to_db)

```

```
61      conn.commit()
62
63
64
65      rt_lmp = conn.execute("SELECT LMP from lmp_data WHERE `Settlement Location`=?",
        (location,))
66
67      rt_lmp = (rt_lmp.fetchone()[0])
68
69      rt_lmp = float(rt_lmp)/1000
70
71
72
73      #print(type(rt_lmp))
74
75      #print("Location: " + location)
76
77      #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
78
79      #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")
80
81
82
83      conn.close()
84
85
86
87      return rt_lmp
88
89
```

```
import requests, csv, io, sqlite3, urllib3

import pandas as pd

from pathlib import Path



urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)



location = 'AEC'



def get_day_ahead_LMP(location):

    folder = Path("C:/Users/Admin")

    db_file = str(folder / "test_10.sqlite")

    url = 'https://marketplace.spp.org/file-api/download/da-lmp-by-location?path=%2F2019%2F04%2FBy_Day%2FDA-LMP-SL-201904290100.csv'



    data = requests.get(url, verify=False)



    df = pd.read_csv(io.StringIO(data.text))

    #print(df.head())



    df.to_csv('day_ahead_LMP.csv')



    conn = sqlite3.connect(db_file)



    conn.execute("DROP TABLE IF EXISTS day_ahead_lmp_data")

    conn.execute("CREATE TABLE day_ahead_lmp_data (Interval, GMTIntervalEnd, `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")



    with open('day_ahead_LMP.csv','r') as fin:

        dict_read = csv.DictReader(fin)

        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'], i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]



    conn.executemany("INSERT INTO day_ahead_lmp_data (Interval, GMTIntervalEnd, `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?, ?);", to_db)
```

```
60
61          conn.commit()
62
63
64
65          da_lmp = conn.execute("SELECT LMP from day_ahead_lmp_data WHERE `Settlement
            Location`=?", (location,))
66
67          da_lmp = (da_lmp.fetchone()[0])
68
69          da_lmp = float(da_lmp)/1000
70
71
72
73          #print(type(da_lmp))
74
75          #print("Location: " + location)
76
77          #print("day ahead LMP (MWh): $" + str(da_lmp) + "/MWh.")
78
79          #print("day ahead LMP (kWh): $" + str(da_lmp/1000) + "/kWh.")
80
81
82
83          #conn.close()
84
85
86
87          return da_lmp
88
89
90
91      get_day_ahead_LMP(location)
```

```
#!/usr/bin/env python3

import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging

from pathlib import Path

from LMP_csv_import import get_realtime_LMP



urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)



folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")

miner_hashrate = 14300000*272

kW_load = 1.3*272*1.05

location = 'AEC'

block_reward = 144



def get_BTC_price():

    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')

    price_dict = price.json()

    BTC_price = price_dict['data']['amount']

    BTC_price = float(BTC_price)

    return BTC_price



def get_network_difficulty():

    network_difficulty =
    requests.get('https://blockexplorer.com/api/status?q=getDifficulty')

    diff_dict = network_difficulty.json()

    diff = diff_dict['difficulty']

    return diff



def get_block_height():

    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')

    bh_dict = block_height.json()



    #print(bh_dict)
```

```
    block_height = bh_dict['info']['blocks']

    block_height = int(block_height)

    print(str(block_height))

    return block_height


def get_network_hashrate(diff):

    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")

    hashrate_dict = network_hashrate.json()

    #print(hashrate_dict)

    hashrate = hashrate_dict['data']['hashrate']

    hashrate = float(hashrate)/1000000

    #hashrate = int(hashrate)

    #print("Called network hashrate: " + str(hashrate))


    expected_blocks = 144

    blocks_found = 113

    est_hashrate = (blocks_found / expected_blocks * diff * 2 ** 32 / 600)

    #print("Estimated network hashrate: " + str(est_hashrate))

    return hashrate


def get_profit(BTC_price, hashrate, lmp_price):

    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)

    print("5m gross USD profit:   $" + str(gross_profit_USD))


    power_cost_USD = kW_load*lmp_price/6/1000

    print("5m electricity cost:   $" + str(power_cost_USD))


    net_profit_USD = gross_profit_USD - power_cost_USD

    print("5m net USD profi/loss: $" + str(net_profit_USD))


    net_profit_BTC = net_profit_USD/BTC_price

    print("5m net BTC profit:     " + str(net_profit_BTC))

```

```
129
130
131         return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
132
133
134
135     def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
136
137         breakeven = ((miner_hashrate/hashrate)*(block_reward*144)*(BTC_price))/(kW_load*24)
138
139         print(breakeven)
140
141         return breakeven
142
143
144
145     def main():
146
147         try:
148
149
150
151             print(str(datetime.datetime.now()))
152
153             block_height = get_block_height()
154
155             BTC_price = get_BTC_price()
156
157             diff = get_network_difficulty()
158
159             hashrate = get_network_hashrate(diff)
160
161             lmp_price = get_realtime_LMP(location)
162
163             breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
                kW_load)
164
165             gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
                hashrate=get_network_hashrate(diff),
166
167                                                         lmp_price=get_realtime_LMP(location))
168
169
170
171             conn = sqlite3.connect(db_file)
172
173             conn.execute("INSERT INTO lmp_model (date_time, BTC_USD_price,
                BTC_network_hashrate, real_time_LMP_mwh, "
174
175                          "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                             net_profit_USD_5m, net_profit_BTC_5m, block_height) "
176
177                          "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                             ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
178
179                                                                   (lmp_price/1000), gp_USD,
                                                                      pc_USD, np_USD, np_BTC,
                                                                      block_height))
180
181             conn.commit()
182
183
184
185         except Exception as e:
186
```

```
            print("Error:  " + str(e))


while True:

    main()

    time.sleep(300)

```

```python
#!/usr/bin/env python3
import sqlite3, smtplib, ssl, socket, json, os, datetime
from pathlib import Path
from toggle_relay import *
from email_alert import *
from LMP_csv_import import get_realtime_LMP

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
restart_max = 2
reboot_max = 1
timeout = .1
threshold = .75
port = 4028
location = 'AEC'

def get_restart_count(user):
    conn = sqlite3.connect(db_file)
    restart_count = conn.execute("SELECT restart_count from miners WHERE miner_name=?", (user,))
    restart_count = (restart_count.fetchone()[0])
    restart_count = str(restart_count)

def update_restart_counter(user):
    conn = sqlite3.connect(db_file)
    restart_count = conn.execute("SELECT restart_count from miners WHERE miner_name=?", (user,))
    restart_count = (restart_count.fetchone()[0])
    restart_count +=1
    restart_count = str(restart_count)

    conn.execute('UPDATE miners SET restart_count =? WHERE miner_name=?', (restart_count, user))
    conn.execute('UPDATE miners SET last_restart_datetime =? WHERE miner_name=?', (str(datetime.datetime.now()), user))
    conn.commit()

    restart_count = conn.execute("SELECT restart_count from miners WHERE miner_name=?", (user,))
    restart_count = (restart_count.fetchone()[0])
    print("Number of restarts: " + str(restart_count))

    email_alert_restart(user, restart_count)

def check_status(host_address):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(timeout)
        sock.connect((host_address, port))

        message = json.dumps({'command': 'lcd'})
        sock.sendto(message.encode(), (host_address, port))
        response = sock.recv(4096).decode("ascii").rstrip(' \t\r\n\0')
        response_dict = json.loads(response)
        print("_____")
        #print(response_dict)

        print("IP: " + host_address)

        user = response_dict['LCD'][0]['User']
        print("Miner: " + str(user))

        GHS_5s = (response_dict['LCD'][0]['GHS 5s'])
        print("5s hashrate: " + (str(GHS_5s/1000)) + " TH/s")

```

```
61          GHS_5m = (response_dict['LCD'][0]['GHS 5m'])
62          print("5m hashrate: " + (str(GHS_5m/1000)) + " TH/s")
63
64          print("Restart threshold: " + (str(GHS_5m*threshold/1000)) + " TH/s")
65
66          uptime = int(response_dict['LCD'][0]['Elapsed'])
67          uptime = str(datetime.timedelta(seconds=uptime))
68          print("Uptime: " + str(uptime))
69
70          temp_celsius = int(response_dict['LCD'][0]['Temperature'])
71          temp_farenheit = int((temp_celsius*(9/5))+ 32)
72          temp_celsius = str(temp_celsius)
73          temp_farenheit = str(temp_farenheit)
74
75          print("Temp: " + temp_celsius + "C/" + temp_farenheit + "F")
76
77          print("_____")
78          sock.shutdown(socket.SHUT_RDWR)
79          sock.close()
80
81          conn = sqlite3.connect(db_file)
82          conn.execute("UPDATE miners SET host_address =? WHERE miner_name=?",
            (host_address, user))
83          conn.execute("UPDATE miners SET miner_hashrate =? WHERE miner_name=?", (GHS_5s,
            user))
84          conn.commit()
85
86          #print("Host Address: " + host_address)
87
88          maintenance = conn.execute("SELECT maintenance from miners WHERE miner_name=?",
            (user,))
89          maintenance = maintenance.fetchone()[0]
90          #print("Maintenace: " + str(maintenance))
91
92          if(maintenance == 1):
93              conn=sqlite3.connect(db_file)
94
95              conn.execute("UPDATE miners SET restart_count = 0, io_reboot_count = 0,
                maintenance = 0 WHERE miner_name=?", (user,))
96              conn.commit()
97
98              email_maintenance_done(user)
99
100         if((int(GHS_5s)) > (int(GHS_5m))): #Starting Logic Here!
101             conn = sqlite3.connect(db_file)
102
103             restart_count = conn.execute("SELECT restart_count from miners WHERE
                miner_name=?", (user,))
104             restart_count = (restart_count.fetchone()[0])
105             print("Restart count: " + str(restart_count))
106
107             io_reboot_count = conn.execute("SELECT io_reboot_count from miners WHERE
                miner_name=?", (user,))
108             io_reboot_count = (io_reboot_count.fetchone()[0])
109             print("Reboot count: " + str(io_reboot_count))
110
111             if(restart_count < restart_max):
112                 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
113                 sock.settimeout(timeout)
114                 sock.connect((host_address, port))
115
116                 message = json.dumps({'command': 'restart'})
117                 sock.sendto(message.encode(), (host_address, port))
118
119                 sock.shutdown(socket.SHUT_RDWR)
```

```
120                 sock.close()
121
122                 #print("Socket closed")
123                 update_restart_counter(user)
124
125             elif(io_reboot_count < reboot_max):
126                 io_reboot_count += 1
127                 io_reboot_count = str(io_reboot_count)
128
129                 conn = sqlite3.connect(db_file)
130                 conn.execute('UPDATE miners SET io_reboot_count =? WHERE miner_name=?',
                    (io_reboot_count, user))
131                 conn.execute('UPDATE miners SET restart_count = 0 WHERE miner_name=?',
                    (user,))
132                 conn.execute('UPDATE miners SET last_reboot_datetime =? WHERE
                    miner_name=?', (str(datetime.datetime.now()), user))
133                 conn.commit()
134
135                 toggle_off_on(user)
136                 email_alert_reboot(user, io_reboot_count)
137
138             else:
139                 toggle_off_repair(user)
140                 email_miner_shutdown(user, io_reboot_count)
141
142     except Exception as e:
143
144             print("Error:  " + host_address + " " + str(e))
145             print("_____________________________________________")
146
147 def main():
148     for i in range(1):
149         #get_realtime_LMP(location)
150         for j in range(25):
151             host_address = "192.168.%d.%d" % (i + 1, j + 1)
152             check_status(host_address)
153 while True:
154     main() #Forever loop through the loop method
155
```

```
from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import smtplib


'''msg = MIMEMultipart()

password = "@Pa$5word123"  # enter your email password here

msg['From'] = "bearbox.catchall@gmail.com"  # from email account

msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring'''




def email_alert_restart(user, restart_count):

    msg = MIMEMultipart()

    password = "@Pa$5word123"  # enter your email password here

    msg['From'] = "bearbox.catchall@gmail.com"  # from email account

    msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring

    message = "The cgminer service has been restarted on:\n\n" + str(user) + ".
    \nRestart Count: " + str(

        restart_count) + "\n\nThis email notification service is provided by the
        BearBox Management Suite."



    msg['Subject'] = "Restart Alert (cgminer) - " + user

    msg.attach(MIMEText(message, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com: 587')

    server.starttls()

    server.login(msg['From'], password)

    server.sendmail(msg['From'], msg['To'], msg.as_string())


    server.quit()




def email_alert_reboot(user, io_reboot_count):

    msg = MIMEMultipart()

    password = "@Pa$5word123"  # enter your email password here

    msg['From'] = "bearbox.catchall@gmail.com"  # from email account
```

```
64
65      msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring
66
67      message = "The power has been restarted on:\n\n" + str(user) + ". \nRestart Count:
        " + str(
68
69          io_reboot_count) + "\n\nThis email notification service is provided by the
            BearBox Management Suite."
70
71      msg['Subject'] = "Reboot Alert (power) - " + user
72
73      msg.attach(MIMEText(message, 'plain'))
74
75      server = smtplib.SMTP('smtp.gmail.com: 587')
76
77      server.starttls()
78
79      server.login(msg['From'], password)
80
81      server.sendmail(msg['From'], msg['To'], msg.as_string())
82
83
84
85      server.quit()
86
87
88
89
90
91  def email_miner_shutdown(user, io_reboot_count):
92
93      msg = MIMEMultipart()
94
95      password = "@Pa$5word123"  # enter your email password here
96
97      msg['From'] = "bearbox.catchall@gmail.com"  # from email account
98
99      msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring
100
101     message = str(user) + " has been permanently shutdown and requires maintenance:
        \nIO Reboots(" + str(
102
103         io_reboot_count) + ")\n\nThis email notification service is provided by the
            BearBox Management Suite."
104
105     msg['Subject'] = "Maintenance Required - " + user
106
107     msg.attach(MIMEText(message, 'plain'))
108
109     server = smtplib.SMTP('smtp.gmail.com: 587')
110
111     server.starttls()
112
113     server.login(msg['From'], password)
114
115     server.sendmail(msg['From'], msg['To'], msg.as_string())
116
117
118
119     server.quit()
120
121
122
123 def email_alert_RT_LMP_miners_off(real_time_LMP, breakeven):
124
```

```
125         msg = MIMEMultipart()
126
127         password = "@Pa$5word123"  # enter your email password here
128
129         msg['From'] = "bearbox.catchall@gmail.com"  # from email account
130
131         msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring
132
133         message = "Miners off.\n\nDA LMP: $" + str(day_ahead_LMP) + "/kWh\nRT LMP: $" +
            str(real_time_LMP) + "/kWh\nBreakeven: $" + str(breakeven) + "/kWh"
134
135         msg['Subject'] = "Miners Are Off!"
136
137         msg.attach(MIMEText(message, 'plain'))
138
139         server = smtplib.SMTP('smtp.gmail.com: 587')
140
141         server.starttls()
142
143         server.login(msg['From'], password)
144
145         server.sendmail(msg['From'], msg['To'], msg.as_string())
146
147
148
149         server.quit()
150
151
152
153     def email_alert_miners_on_LMP(day_ahead_LMP, real_time_LMP, breakeven):
154
155         msg = MIMEMultipart()
156
157         password = "@Pa$5word123"  # enter your email password here
158
159         msg['From'] = "bearbox.catchall@gmail.com"  # from email account
160
161         msg['To'] = "bearbox.catchall@gmail.com"  # to email account for monitoring
162
163         message = "Miners on.\n\nDA LMP: $" + str(day_ahead_LMP) + "/kWh\nRT LMP: $" +
            str(real_time_LMP) + "/kWh\nBreakeven: $" + str(breakeven) + "/kWh"
164
165         msg['Subject'] = "Miners Are On!"
166
167         msg.attach(MIMEText(message, 'plain'))
168
169         server = smtplib.SMTP('smtp.gmail.com: 587')
170
171         server.starttls()
172
173         server.login(msg['From'], password)
174
175         server.sendmail(msg['From'], msg['To'], msg.as_string())
176
177
178
179         server.quit()
```

```
#!/usr/bin/env python3

import json, sqlite3, datetime, time

from datetime import datetime

from pathlib import Path

from bitcoin.rpc import RawProxy



folder = Path("C:/Users/Admin") # this is your path to the sqlite3 db

db_file = str(folder / "test_10.sqlite")  # hover mouse over db creation for URI


def get_block_height_local():

    try:

        p = RawProxy() # connect to node

        info = p.getblockchaininfo() # getblockchaininfo call

        block_height = (info['blocks']) # gets block height from getblockchaininfo



        header_hash = (info['bestblockhash']) # gets the bestblockhash from your node



        header = p.getblock(header_hash) # calls getblock and passes in the header_hash
        param from above



        block_time = header['time'] # unix timestamp of node's best block header

        block_time_utc = datetime.utcfromtimestamp(block_time).strftime('%Y-%m-%d
        %H:%M:%S') # convert unix timestamp to utc



        time_now = time.time() # call time.time() and get unix timestamp of current time

        time_now_utc = datetime.utcfromtimestamp(time_now).strftime('%Y-%m-%d
        %H:%M:%S') # convert current time unix timestamp to utc



        '''prop_time = (time_now - block_time) # calculate  the difference between
        current time and best block time

        prop_time_utc = datetime.utcfromtimestamp(prop_time).strftime('%H:%M:%S') #
        convert unix propagation time to utc in hours/minutes/seconds format'''



        conn = sqlite3.connect(db_file) # connect to sqlite3 db ---- see path and file
        under imports

        conn.execute("INSERT INTO block_height_data_test (block_height,
```

```
        header_block_time, date_time) '''# insert into table name
        'block_height_data_test', column names, values
60
61                      "VALUES (?, ?, ?);", (block_height, block_time_utc, time_now_utc))
62
63          conn.commit()
64
65
66
67      except Exception as e: # catch exceptions - specifically, block_height has a unique
        constraint in the created sqlite3 table
68
69              print("Error:  " + str(e))
70
71
72
73  while True:
74
75      get_block_height_local() # call the above
76
77      time.sleep(.1) # delay .1 seconds in while loop
```

```
#!/usr/bin/env python3

import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time

from pathlib import Path

from LMP_csv_import import get_realtime_LMP


urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) #turns off SSL certificate warning from requesting LMP Data CSV


folder = Path("C:/Users/Admin")

db_file = str(folder / "test_10.sqlite")

miner_hashrate = 14500000*272*.95

kW_load = 1.3*272*1.05

location = 'AEC'


def get_BTC_price():

    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')

    price_dict = price.json()

    #print(price_dict)

    BTC_price = price_dict['data']['amount']

    BTC_price = float(BTC_price)

    #print("BTC/USD price   : $" + str(BTC_price))


    return BTC_price


def get_network_difficulty():

    network_difficulty = requests.get('https://blockexplorer.com/api/status?q=getDifficulty')

    diff_dict = network_difficulty.json()

    #print(diff_dict)

    diff = diff_dict['difficulty']

    #print("Difficulty: " + str(diff))

    return diff


def get_block_height():
```

```
64
65      block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
66
67      bh_dict = block_height.json()
68
69
70
71      #print(bh_dict)
72
73      block_height = bh_dict['info']['blocks']
74
75      block_height = int(block_height)
76
77      print(str(block_height))
78
79      return block_height
80
81
82
83  def get_network_hashrate(diff):
84
85      network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
86
87      hashrate_dict = network_hashrate.json()
88
89      #print(hashrate_dict)
90
91      hashrate = hashrate_dict['data']['hashrate']
92
93      hashrate = float(hashrate)/1000000
94
95      return hashrate
96
97
98
99  def get_profit(BTC_price, hashrate, lmp_price):
100
101     gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
102
103     print("5m gross USD profit:   $" + str(gross_profit_USD))
104
105
106
107     power_cost_USD = kW_load*lmp_price/6/1000
108
109     print("5m electricity cost:   $" + str(power_cost_USD))
110
111
112
113     net_profit_USD = gross_profit_USD - power_cost_USD
114
115     print("5m net USD profi/loss: $" + str(net_profit_USD))
116
117
118
119     net_profit_BTC = net_profit_USD/BTC_price
120
121     print("5m net BTC profit:     " + str(net_profit_BTC))
122
123
124
125     return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
126
127
128
```

```
129
130
131 def main():
132
133     try:
134
135         print(str(datetime.datetime.now()))
136
137         block_height = get_block_height()
138
139         BTC_price = get_BTC_price()
140
141         diff = get_network_difficulty()
142
143         hashrate = get_network_hashrate(diff)
144
145         lmp_price = get_realtime_LMP(location)
146
147         gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
148
149                                                     lmp_price=get_realtime_LMP(location))
150
151
152
153         conn = sqlite3.connect(db_file)
154
155         conn.execute("INSERT INTO lmp_model_effi (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
156
157                      "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
158
159                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
160
161                                                             (lmp_price/1000), gp_USD,
                                                                pc_USD, np_USD, np_BTC,
                                                                block_height))
162
163         conn.commit()
164
165
166
167     except Exception as e:
168
169             print("Error:  " + str(e))
170
171
172
173 while True:
174
175     try:
176
177         main()
178
179         time.sleep(300)
180
181     except:
182
183         main()
184
185         time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time
from pathlib import Path
from LMP_csv_import import get_realtime_LMP

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000*272
kW_load = 1.3*272*1.05
location = 'AEC'

def get_BTC_price():
    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
    price_dict = price.json()
    BTC_price = price_dict['data']['amount']
    BTC_price = float(BTC_price)
    return BTC_price

def get_network_difficulty():
    network_difficulty =
    requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
    diff_dict = network_difficulty.json()
    diff = diff_dict['difficulty']
    return diff

def get_block_height():
    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
    bh_dict = block_height.json()

    #print(bh_dict)
    block_height = bh_dict['info']['blocks']
    block_height = int(block_height)
    print(str(block_height))
    return block_height

def get_network_hashrate(diff):
    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
    hashrate_dict = network_hashrate.json()
    #print(hashrate_dict)
    hashrate = hashrate_dict['data']['hashrate']
    hashrate = float(hashrate)/1000000
    #hashrate = int(hashrate)
    #print("Called network hashrate: " + str(hashrate))

    expected_blocks = 144
    blocks_found = 113
    est_hashrate = (blocks_found / expected_blocks * diff * 2 ** 32 / 600)
    #print("Estimated network hashrate: " + str(est_hashrate))
    return hashrate

def get_profit(BTC_price, hashrate, lmp_price):
    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
    print("5m gross USD profit:   $" + str(gross_profit_USD))

    power_cost_USD = kW_load*lmp_price/6/1000
    print("5m electricity cost:   $" + str(power_cost_USD))

    net_profit_USD = gross_profit_USD - power_cost_USD
    print("5m net USD profi/loss: $" + str(net_profit_USD))

    net_profit_BTC = net_profit_USD/BTC_price
    print("5m net BTC profit:     " + str(net_profit_BTC))

```

```
65      return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
66
67
68  def main():
69      try:
70          print(str(datetime.datetime.now()))
71          block_height = get_block_height()
72          BTC_price = get_BTC_price()
73          diff = get_network_difficulty()
74          hashrate = get_network_hashrate(diff)
75          lmp_price = get_realtime_LMP(location)
76          gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
77                                                      lmp_price=get_realtime_LMP(location))
78
79          conn = sqlite3.connect(db_file)
80          conn.execute("INSERT INTO lmp_model (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
81                       "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
82                       "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
83                                                              (lmp_price/1000), gp_USD,
                                                                pc_USD, np_USD, np_BTC,
                                                                block_height))
84          conn.commit()
85
86
87  while True:
88      main()
89      time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON4'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if(day_ahead_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
            return
        elif(real_time_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
            return
        else:
            from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
                email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)
            else:
                return

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))
```

```
129
130 def main():
131     try:
132         block_height = get_block_height()
133         print("Current Block Height:          " + str(block_height))
134
135         BTC_price = get_BTC_price()
136         BTC_price = float("{0:.2f}".format(BTC_price))
137         print("Current BTC Price:             " + str(BTC_price))
138
139         diff = get_network_difficulty()
140         print("Network difficulty:            " + str(diff))
141
142         hashrate = get_network_hashrate(diff)
143         print("Estimated Network Hashrate:    " + str(hashrate))
144
145         day_ahead_LMP = get_day_ahead_LMP(location)
146         day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
147         print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
148
149         real_time_LMP = get_real_time_LMP(location)
150         real_time_LMP = float("{0:.7f}".format(real_time_LMP))
151         print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
152
153         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
154         breakeven = float("{0:.7f}".format(breakeven))
155         print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
156         print("_____________________________________________________________")
157
158         profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
159
160         day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)
161
162         if(day_ahead_LMP >= breakeven):
163             realized_rev = day_ahead_LMP_rev
164         elif(real_time_LMP >= breakeven):
165             realized_rev = real_time_LMP_rev
166         else:
167             realized_rev = mining_rev
168
169         conn = sqlite3.connect(db_file)
170
171         conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON4 (datetime,
        block_height,"
172                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev)")
173
174         conn.execute("INSERT INTO arb_main_EXELON4 (datetime, block_height,"
175                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev) "
176                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
177                      (str(datetime.datetime.now()), block_height,
178                       diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                          breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                          realized_rev))
179         conn.commit()
180
181     except Exception as e:
182         print("Error:  " + str(e))
183
184 while True:
```

```
185         main()
186         time.sleep(300)
187
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON_HPW1'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON_HPW1 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON_HPW1 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON4'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if(day_ahead_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif(real_time_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
                email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven, location)
            else:
                return

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))
```

```
129
130 def main():
131     try:
132         block_height = get_block_height()
133         print("Current Block Height:          " + str(block_height))
134
135         BTC_price = get_BTC_price()
136         BTC_price = float("{0:.2f}".format(BTC_price))
137         print("Current BTC Price:             " + str(BTC_price))
138
139         diff = get_network_difficulty()
140         print("Network difficulty:            " + str(diff))
141
142         hashrate = get_network_hashrate(diff)
143         print("Estimated Network Hashrate:    " + str(hashrate))
144
145         day_ahead_LMP = get_day_ahead_LMP(location)
146         day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
147         print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
148
149         real_time_LMP = get_real_time_LMP(location)
150         real_time_LMP = float("{0:.7f}".format(real_time_LMP))
151         print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
152
153         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
            kW_load)
154         breakeven = float("{0:.7f}".format(breakeven))
155         print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
156         print("_____________________________________________________________")
157
158         profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
159
160         day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
            hashrate, real_time_LMP, day_ahead_LMP)
161
162         if(day_ahead_LMP >= breakeven):
163             realized_rev = day_ahead_LMP_rev
164         elif(real_time_LMP >= breakeven):
165             realized_rev = real_time_LMP_rev
166         else:
167             realized_rev = mining_rev
168
169         conn = sqlite3.connect(db_file)
170
171         conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON4 (datetime,
            block_height,"
172                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev)")
173
174         conn.execute("INSERT INTO arb_main_EXELON4 (datetime, block_height,"
175                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev) "
176                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
177                      (str(datetime.datetime.now()), block_height,
178                       diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                          breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                          realized_rev))
179         conn.commit()
180
181     except Exception as e:
182         print("Error:  " + str(e))
183
184 while True:
```

```
185         main()
186         time.sleep(300)
187
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON5_6'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON5_6 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON5_6 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON7_8'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON7_8 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON7_8 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON9'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON9 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON9 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON10_11'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
65          breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
            (kW_load * 24)
66          return breakeven
67
68      except Exception as e:
69          print("Error:  " + str(e))
70
71  def pdu1_all_on():
72      try:
73          from pymodbus.client.sync import ModbusTcpClient as ModbusClient
74          client = ModbusClient(config.pdu1_ip, config.pdu1_port)
75          client.write_coils(0, [False] * 24)
76          client.close()
77
78      except Exception as e:
79          print("Error:  " + str(e))
80
81  def pdu1_all_off():
82      try:
83          from pymodbus.client.sync import ModbusTcpClient as ModbusClient
84          client = ModbusClient(config.pdu1_ip, config.pdu1_port)
85          client.write_coils(0, [True]*24)
86          client.close()
87
88      except Exception as e:
89          print("Error:  " + str(e))
90
91  def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
92      try:
93          if (day_ahead_LMP >= breakeven):
94              '''pdu1_all_off()'''
95              email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
96              return
97          elif (real_time_LMP >= breakeven):
98              '''pdu1_all_off()'''
99              email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
100             return
101         else:
102             '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
103             client = ModbusClient(config.pdu1_ip, config.pdu1_port)
104             r = client.read_coils(0, 24)
105             if(r.bits == [True]*24):
106                 pdu1_all_on()
107             email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''
108
109     except Exception as e:
110         print("Error:  " + str(e))
111
112 def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
113     try:
114         day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
115         day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))
116
117         real_time_LMP_rev = real_time_LMP * kW_load/(12)
118         real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))
119
120         mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
121         mining_rev = float("{0:.7f}".format(mining_rev))
122
123         return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev
124
125     except Exception as e:
126         print("Error:  " + str(e))
127
128 def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON10_11 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON10_11 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
import requests, csv, io, sqlite3, urllib3, datetime
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_day_ahead_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")

    now = datetime.datetime.now()
    #print(str(now))

    year = now.strftime("%Y")
    month = now.strftime("%m")
    day = now.strftime("%d")

    '''if(month == '04' OR month == '06' OR month == '09' OR month == '11'):'''

    url = 'https://marketplace.spp.org/file-api/download/da-lmp-by-location?path=%2F' +
    year + '%2F' + month + '%2FBy_Day%2FDA-LMP-SL-' + year + month + day + '0100.csv'
    #print(url)

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))

    df.to_csv('day_ahead_LMP.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS day_ahead_lmp_data_AEC")
    conn.execute("CREATE TABLE day_ahead_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('day_ahead_LMP.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO day_ahead_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    da_lmp = conn.execute("SELECT LMP from day_ahead_lmp_data WHERE `Settlement
    Location`=?", (location,))
    da_lmp = (da_lmp.fetchone()[0])
    da_lmp = float(da_lmp)/1000

    return da_lmp
```

```
import requests, csv, io, sqlite3, urllib3, datetime
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_day_ahead_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")

    now = datetime.datetime.now()
    #print(str(now))

    year = now.strftime("%Y")
    month = now.strftime("%m")
    day = now.strftime("%d")

    url = 'https://marketplace.spp.org/file-api/download/da-lmp-by-location?path=%2F' +
    year + '%2F' + month + '%2FBy_Day%2FDA-LMP-SL-' + year + month + day + '0100.csv'
    #print(url)

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))

    df.to_csv('day_ahead_LMP.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS day_ahead_lmp_data_AEC")
    conn.execute("CREATE TABLE day_ahead_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('day_ahead_LMP.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO day_ahead_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    da_lmp = conn.execute("SELECT LMP from day_ahead_lmp_data_AEC WHERE `Settlement
    Location`=?", (location,))
    da_lmp = (da_lmp.fetchone()[0])
    da_lmp = float(da_lmp)/1000

    return da_lmp
```

```python
#!/usr/bin/env python3
import json, sqlite3, datetime, time
from datetime import datetime
from pathlib import Path
from bitcoin.rpc import RawProxy

folder = Path("C:/Users/Admin") # this is your path to the sqlite3 db
db_file = str(folder / "test_10.sqlite")  # hover mouse over db creation for URI

def get_block_height_local():
    try:
        p = RawProxy() # connect to node
        info = p.getblockchaininfo() # getblockchaininfo call
        block_height = (info['blocks']) # gets block height from getblockchaininfo

        header_hash = (info['bestblockhash']) # gets the bestblockhash from your node

        header = p.getblock(header_hash) # calls getblock and passes in the header_hash param from above

        block_time = header['time'] # unix timestamp of node's best block header
        block_time_utc = datetime.utcfromtimestamp(block_time).strftime('%Y-%m-%d %H:%M:%S') # convert unix timestamp to utc

        time_now = time.time() # call time.time() and get unix timestamp of current time
        time_now_utc = datetime.utcfromtimestamp(time_now).strftime('%Y-%m-%d %H:%M:%S') # convert current time unix timestamp to utc

        '''prop_time = (time_now - block_time) # calculate  the difference between current time and best block time
        prop_time_utc = datetime.utcfromtimestamp(prop_time).strftime('%H:%M:%S') # convert unix propagation time to utc in hours/minutes/seconds format'''

        conn = sqlite3.connect(db_file) # connect to sqlite3 db ---- see path and file under imports
        conn.execute("INSERT INTO block_height_data_test (block_height, header_block_time, date_time) " # insert into table name 'block_height_data_test', column names, values
                     "VALUES (?, ?, ?);", (block_height, block_time_utc, time_now_utc))
        conn.commit()

    except Exception as e: # catch exceptions - specifically, block_height has a unique constraint in the created sqlite3 table
            print("Error:  " + str(e))

while True:
    get_block_height_local() # call the above
    time.sleep(.1) # delay .1 seconds in while loop
```

```python
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url = 'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data_AEC")
    conn.execute("CREATE TABLE real_time_lmp_data_AEC (Interval, GMTIntervalEnd, `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'], i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data_AEC (Interval, GMTIntervalEnd, `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?, ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data_AEC WHERE `Settlement Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)



def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data_EXELON4")
    conn.execute("CREATE TABLE real_time_lmp_data_EXELON4 (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data_EXELON4 (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data_EXELON4 WHERE `Settlement
    Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

get_real_time_LMP(location)

```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data")
    conn.execute("CREATE TABLE real_time_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data WHERE `Settlement
    Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging
from pathlib import Path
from LMP_csv_import import get_realtime_LMP

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000*272
kW_load = 1.3*272*1.05
location = 'AEC'
block_reward = 144

def get_BTC_price():
    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
    price_dict = price.json()
    BTC_price = price_dict['data']['amount']
    BTC_price = float(BTC_price)
    return BTC_price

def get_network_difficulty():
    network_difficulty = requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
    diff_dict = network_difficulty.json()
    diff = diff_dict['difficulty']
    return diff

def get_block_height():
    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
    bh_dict = block_height.json()

    #print(bh_dict)
    block_height = bh_dict['info']['blocks']
    block_height = int(block_height)
    print(str(block_height))
    return block_height

def get_network_hashrate(diff):
    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
    hashrate_dict = network_hashrate.json()
    #print(hashrate_dict)
    hashrate = hashrate_dict['data']['hashrate']
    hashrate = float(hashrate)/1000000
    #hashrate = int(hashrate)
    #print("Called network hashrate: " + str(hashrate))

    expected_blocks = 144
    blocks_found = 113
    est_hashrate = (blocks_found / expected_blocks * diff * 2 ** 32 / 600)
    #print("Estimated network hashrate: " + str(est_hashrate))
    return hashrate

def get_profit(BTC_price, hashrate, lmp_price):
    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
    print("5m gross USD profit:   $" + str(gross_profit_USD))

    power_cost_USD = kW_load*lmp_price/6/1000
    print("5m electricity cost:   $" + str(power_cost_USD))

    net_profit_USD = gross_profit_USD - power_cost_USD
    print("5m net USD profi/loss: $" + str(net_profit_USD))

    net_profit_BTC = net_profit_USD/BTC_price
```

```
64      print("5m net BTC profit:     " + str(net_profit_BTC))
65
66      return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
67
68  def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
69      breakeven = ((miner_hashrate/hashrate)*(block_reward*144)*(BTC_price))/(kW_load*24)
70      print(breakeven)
71      return breakeven
72
73  def main():
74      try:
75
76          print(str(datetime.datetime.now()))
77          block_height = get_block_height()
78          BTC_price = get_BTC_price()
79          diff = get_network_difficulty()
80          hashrate = get_network_hashrate(diff)
81          lmp_price = get_realtime_LMP(location)
82          breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
            kW_load)
83          gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
84                                                      lmp_price=get_realtime_LMP(location))
85
86          conn = sqlite3.connect(db_file)
87          conn.execute("INSERT INTO lmp_model (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
88                       "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
89                       "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
90                                                              (lmp_price/1000), gp_USD,
                                                                pc_USD, np_USD, np_BTC,
                                                                block_height))
91          conn.commit()
92
93      except Exception as e:
94              print("Error:  " + str(e))
95
96  while True:
97      main()
98      time.sleep(300)
99
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time
from pathlib import Path
from LMP_csv_import import get_realtime_LMP

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) #turns off SSL certificate warning from requesting LMP Data CSV

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14500000*272*.95
kW_load = 1.3*272*1.05
location = 'AEC'

def get_BTC_price():
    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
    price_dict = price.json()
    #print(price_dict)
    BTC_price = price_dict['data']['amount']
    BTC_price = float(BTC_price)
    #print("BTC/USD price   : $" + str(BTC_price))

    return BTC_price

def get_network_difficulty():
    network_difficulty = requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
    diff_dict = network_difficulty.json()
    #print(diff_dict)
    diff = diff_dict['difficulty']
    #print("Difficulty: " + str(diff))
    return diff

def get_block_height():
    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
    bh_dict = block_height.json()

    #print(bh_dict)
    block_height = bh_dict['info']['blocks']
    block_height = int(block_height)
    print(str(block_height))
    return block_height

def get_network_hashrate(diff):
    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
    hashrate_dict = network_hashrate.json()
    #print(hashrate_dict)
    hashrate = hashrate_dict['data']['hashrate']
    hashrate = float(hashrate)/1000000
    return hashrate

def get_profit(BTC_price, hashrate, lmp_price):
    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
    print("5m gross USD profit:   $" + str(gross_profit_USD))

    power_cost_USD = kW_load*lmp_price/6/1000
    print("5m electricity cost:   $" + str(power_cost_USD))

    net_profit_USD = gross_profit_USD - power_cost_USD
    print("5m net USD profi/loss: $" + str(net_profit_USD))

    net_profit_BTC = net_profit_USD/BTC_price
    print("5m net BTC profit:     " + str(net_profit_BTC))

    return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
```

```
64
65
66  def main():
67      try:
68          print(str(datetime.datetime.now()))
69          block_height = get_block_height()
70          BTC_price = get_BTC_price()
71          diff = get_network_difficulty()
72          hashrate = get_network_hashrate(diff)
73          lmp_price = get_realtime_LMP(location)
74          gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
75                                                      lmp_price=get_realtime_LMP(location))
76
77          conn = sqlite3.connect(db_file)
78          conn.execute("INSERT INTO lmp_model_effi (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
79                       "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
80                       "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
81                                                               (lmp_price/1000), gp_USD,
                                                                 pc_USD, np_USD, np_BTC,
                                                                 block_height))
82          conn.commit()
83
84      except Exception as e:
85              print("Error:  " + str(e))
86
87  while True:
88      try:
89          main()
90          time.sleep(300)
91      except:
92          main()
93          time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON_HPW1'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON_HPW1 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON_HPW1 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time
from pathlib import Path
from LMP_csv_import import get_realtime_LMP

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning) #turns off SSL certificate warning from requesting LMP Data CSV

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14500000*272*.95
kW_load = 1.3*272*1.05
location = 'AEC'

def get_BTC_price():
    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
    price_dict = price.json()
    #print(price_dict)
    BTC_price = price_dict['data']['amount']
    BTC_price = float(BTC_price)
    #print("BTC/USD price   : $" + str(BTC_price))

    return BTC_price

def get_network_difficulty():
    network_difficulty = requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
    diff_dict = network_difficulty.json()
    #print(diff_dict)
    diff = diff_dict['difficulty']
    #print("Difficulty: " + str(diff))
    return diff

def get_block_height():
    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
    bh_dict = block_height.json()

    #print(bh_dict)
    block_height = bh_dict['info']['blocks']
    block_height = int(block_height)
    print(str(block_height))
    return block_height

def get_network_hashrate(diff):
    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
    hashrate_dict = network_hashrate.json()
    #print(hashrate_dict)
    hashrate = hashrate_dict['data']['hashrate']
    hashrate = float(hashrate)/1000000
    return hashrate

def get_profit(BTC_price, hashrate, lmp_price):
    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
    print("5m gross USD profit:   $" + str(gross_profit_USD))

    power_cost_USD = kW_load*lmp_price/6/1000
    print("5m electricity cost:   $" + str(power_cost_USD))

    net_profit_USD = gross_profit_USD - power_cost_USD
    print("5m net USD profi/loss: $" + str(net_profit_USD))

    net_profit_BTC = net_profit_USD/BTC_price
    print("5m net BTC profit:     " + str(net_profit_BTC))

    return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
```

```
64
65
66  def main():
67      try:
68          print(str(datetime.datetime.now()))
69          block_height = get_block_height()
70          BTC_price = get_BTC_price()
71          diff = get_network_difficulty()
72          hashrate = get_network_hashrate(diff)
73          lmp_price = get_realtime_LMP(location)
74          gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
75                                                      lmp_price=get_realtime_LMP(location))
76
77          conn = sqlite3.connect(db_file)
78          conn.execute("INSERT INTO lmp_model_effi (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
79                       "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
80                       "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
81                                                              (lmp_price/1000), gp_USD,
                                                                pc_USD, np_USD, np_BTC,
                                                                block_height))
82          conn.commit()
83
84      except Exception as e:
85              print("Error:  " + str(e))
86
87  while True:
88      try:
89          main()
90          time.sleep(300)
91      except:
92          main()
93          time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON5_6'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON5_6 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON5_6 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON4'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if(day_ahead_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif(real_time_LMP >= breakeven):
            pdu1_all_off()
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
                email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven, location)
            else:
                return

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))
```

```
129
130 def main():
131     try:
132         block_height = get_block_height()
133         print("Current Block Height:          " + str(block_height))
134
135         BTC_price = get_BTC_price()
136         BTC_price = float("{0:.2f}".format(BTC_price))
137         print("Current BTC Price:             " + str(BTC_price))
138
139         diff = get_network_difficulty()
140         print("Network difficulty:            " + str(diff))
141
142         hashrate = get_network_hashrate(diff)
143         print("Estimated Network Hashrate:    " + str(hashrate))
144
145         day_ahead_LMP = get_day_ahead_LMP(location)
146         day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
147         print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
148
149         real_time_LMP = get_real_time_LMP(location)
150         real_time_LMP = float("{0:.7f}".format(real_time_LMP))
151         print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
152
153         breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
154         breakeven = float("{0:.7f}".format(breakeven))
155         print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
156         print("_____________________________________________________________")
157
158         profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
159
160         day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)
161
162         if(day_ahead_LMP >= breakeven):
163             realized_rev = day_ahead_LMP_rev
164         elif(real_time_LMP >= breakeven):
165             realized_rev = real_time_LMP_rev
166         else:
167             realized_rev = mining_rev
168
169         conn = sqlite3.connect(db_file)
170
171         conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON4 (datetime,
        block_height,"
172                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev)")
173
174         conn.execute("INSERT INTO arb_main_EXELON4 (datetime, block_height,"
175                      "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                         real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                         real_time_LMP_rev, mining_rev, realized_rev) "
176                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
177                      (str(datetime.datetime.now()), block_height,
178                       diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                         breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                         realized_rev))
179         conn.commit()
180
181     except Exception as e:
182         print("Error:  " + str(e))
183
184 while True:
```

```
185         main()
186         time.sleep(300)
187
```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)



def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data_EXELON4")
    conn.execute("CREATE TABLE real_time_lmp_data_EXELON4 (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data_EXELON4 (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data_EXELON4 WHERE `Settlement
    Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

get_real_time_LMP(location)

```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from RT_LMP_import import get_real_time_LMP
from DA_LMP_import import get_day_ahead_LMP
from toggle_relay import *
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272
kW_load = 1.3 * 272
location = 'AEC'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        # print(hashrate_dict)
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        # hashrate = int(hashrate)
        # print("Called network hashrate: " + str(hashrate))
        return hashrate

    except Exception as e:
```

```
65          print("Error:  " + str(e))
66
67  def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
68      try:
69          breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
            (kW_load * 24)
70          #print(breakeven)
71          return breakeven
72
73      except Exception as e:
74          print("Error:  " + str(e))
75
76  def pdu1_all_on():
77      try:
78          from pymodbus.client.sync import ModbusTcpClient as ModbusClient
79          client = ModbusClient(config.pdu1_ip, config.pdu1_port)
80          client.write_coils(0, [False] * 24)
81          client.close()
82
83      except Exception as e:
84          print("Error:  " + str(e))
85
86  def pdu1_all_off():
87      try:
88          from pymodbus.client.sync import ModbusTcpClient as ModbusClient
89          client = ModbusClient(config.pdu1_ip, config.pdu1_port)
90          client.write_coils(0, [True]*24)
91          client.close()
92
93      except Exception as e:
94          print("Error:  " + str(e))
95
96  def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
97      try:
98          if(day_ahead_LMP >= breakeven):
99              pdu1_all_off()
100             email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
101             return
102         elif(real_time_LMP >= breakeven):
103             pdu1_all_off()
104             email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
105             return
106         else:
107             from pymodbus.client.sync import ModbusTcpClient as ModbusClient
108             client = ModbusClient(config.pdu1_ip, config.pdu1_port)
109             r = client.read_coils(0, 24)
110             if(r.bits == [True]*24):
111                 pdu1_all_on()
112                 email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)
113             else:
114                 return
115
116     except Exception as e:
117         print("Error:  " + str(e))
118
119 def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
120     try:
121         day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
122         day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))
123
124         real_time_LMP_rev = real_time_LMP * kW_load/(12)
125         real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))
126
127         mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
128         mining_rev = float("{0:.7f}".format(mining_rev))
```

```
129
130             return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev
131
132         except Exception as e:
133             print("Error:  " + str(e))
134
135     def main():
136         try:
137             block_height = get_block_height()
138             print("Current Block Height:          " + str(block_height))
139
140             BTC_price = get_BTC_price()
141             BTC_price = float("{0:.2f}".format(BTC_price))
142             print("Current BTC Price:             " + str(BTC_price))
143
144             diff = get_network_difficulty()
145             print("Network difficulty:            " + str(diff))
146
147             hashrate = get_network_hashrate(diff)
148             print("Estimated Network Hashrate:    " + str(hashrate))
149
150             day_ahead_LMP = get_day_ahead_LMP(location)
151             day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
152             print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
153
154             real_time_LMP = get_real_time_LMP(location)
155             real_time_LMP = float("{0:.7f}".format(real_time_LMP))
156             print("Real Time LMP ($/kWh):         " + str(real_time_LMP))
157
158             breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
                kW_load)
159             breakeven = float("{0:.7f}".format(breakeven))
160             print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
161             print("_____________________________________________________________")
162
163             profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
164
165             day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
                hashrate, real_time_LMP, day_ahead_LMP)
166
167             if(day_ahead_LMP >= breakeven):
168                 realized_rev = day_ahead_LMP_rev
169             elif(real_time_LMP >= breakeven):
170                 realized_rev = real_time_LMP_rev
171             else:
172                 realized_rev = mining_rev
173
174             conn = sqlite3.connect(db_file)
175             conn.execute("INSERT INTO denis_logic_test (datetime, day_ahead_LMP,
                block_height, BTC_price, "
176                          "network_diff, est_network_hashrate, real_time_LMP,
                         breakeven_mining_cost, day_ahead_LMP_rev, real_time_LMP_rev,
                         mining_rev, realized_rev) "
177                          "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
178                          (str(datetime.datetime.now()), day_ahead_LMP, block_height,
                         BTC_price,
179                           diff, hashrate, real_time_LMP, breakeven, day_ahead_LMP_rev,
                          real_time_LMP_rev, mining_rev, realized_rev))
180             conn.commit()
181
182         except Exception as e:
183             print("Error:  " + str(e))
184
185     while True:
186         main()
```

```
187         time.sleep(300)
188
```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data")
    conn.execute("CREATE TABLE real_time_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data WHERE `Settlement
    Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

location = 'AEC'

def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS lmp_data")
    conn.execute("CREATE TABLE lmp_data (Interval, GMTIntervalEnd, `Settlement
    Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO lmp_data (Interval, GMTIntervalEnd, `Settlement
    Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?, ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from lmp_data WHERE `Settlement Location`=?",
    (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

```

```
import requests, csv, io, sqlite3, urllib3
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_real_time_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")
    url =
    'https://marketplace.spp.org/file-api/download/rtbm-lmp-by-location?path=%2FRTBM-LMP-SL-latestInterval.csv'

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))
    #print(df.head())

    df.to_csv('data.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS real_time_lmp_data_AEC")
    conn.execute("CREATE TABLE real_time_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('data.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO real_time_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    rt_lmp = conn.execute("SELECT LMP from real_time_lmp_data_AEC WHERE `Settlement
    Location`=?", (location,))
    rt_lmp = (rt_lmp.fetchone()[0])
    rt_lmp = float(rt_lmp)/1000

    #print(type(rt_lmp))
    #print("Location: " + location)
    #print("Real time LMP (MWh): $" + str(rt_lmp) + "/MWh.")
    #print("Real time LMP (kWh): $" + str(rt_lmp/1000) + "/kWh.")

    conn.close()

    return rt_lmp

```

```
import requests, csv, io, sqlite3, urllib3, datetime
import pandas as pd
from pathlib import Path

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def get_day_ahead_LMP(location):
    folder = Path("C:/Users/Admin")
    db_file = str(folder / "test_10.sqlite")

    now = datetime.datetime.now()
    #print(str(now))

    year = now.strftime("%Y")
    month = now.strftime("%m")
    day = now.strftime("%d")

    '''if(month == '04' OR month == '06' OR month == '09' OR month == '11'):'''

    url = 'https://marketplace.spp.org/file-api/download/da-lmp-by-location?path=%2F' +
    year + '%2F' + month + '%2FBy_Day%2FDA-LMP-SL-' + year + month + day + '0100.csv'
    #print(url)

    data = requests.get(url, verify=False)

    df = pd.read_csv(io.StringIO(data.text))

    df.to_csv('day_ahead_LMP.csv')

    conn = sqlite3.connect(db_file)

    conn.execute("DROP TABLE IF EXISTS day_ahead_lmp_data_AEC")
    conn.execute("CREATE TABLE day_ahead_lmp_data_AEC (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC);")

    with open('day_ahead_LMP.csv','r') as fin:
        dict_read = csv.DictReader(fin)
        to_db = [(i['Interval'], i['GMTIntervalEnd'], i['Settlement Location'],
        i['Pnode'], i['LMP'], i['MLC'], i['MCC'], i['MEC']) for i in dict_read]

    conn.executemany("INSERT INTO day_ahead_lmp_data (Interval, GMTIntervalEnd,
    `Settlement Location`, Pnode, LMP, MLC, MCC, MEC) VALUES (?, ?, ?, ?, ?, ?, ?,
    ?);", to_db)
    conn.commit()

    da_lmp = conn.execute("SELECT LMP from day_ahead_lmp_data WHERE `Settlement
    Location`=?", (location,))
    da_lmp = (da_lmp.fetchone()[0])
    da_lmp = float(da_lmp)/1000

    return da_lmp
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON7_8'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON7_8 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON7_8 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON9'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON9 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON9 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time,
logging, config
from pathlib import Path
from get_current_RT_LMP import get_real_time_LMP
from get_current_DA_LMP import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from bitcoin.rpc import RawProxy
from email_alert import *

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000 * 272 * .95
kW_load = 1.3 * 272 * 1.05
location = 'EXELON10_11'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))

def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
    try:
```

```
        breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
        (kW_load * 24)
        return breakeven

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_on():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [False] * 24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def pdu1_all_off():
    try:
        from pymodbus.client.sync import ModbusTcpClient as ModbusClient
        client = ModbusClient(config.pdu1_ip, config.pdu1_port)
        client.write_coils(0, [True]*24)
        client.close()

    except Exception as e:
        print("Error:  " + str(e))

def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
    try:
        if (day_ahead_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        elif (real_time_LMP >= breakeven):
            '''pdu1_all_off()'''
            email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP, location)
            return
        else:
            '''from pymodbus.client.sync import ModbusTcpClient as ModbusClient
            client = ModbusClient(config.pdu1_ip, config.pdu1_port)
            r = client.read_coils(0, 24)
            if(r.bits == [True]*24):
                pdu1_all_on()
            email_alert_miners_on(day_ahead_LMP, real_time_LMP, breakeven)'''

    except Exception as e:
        print("Error:  " + str(e))

def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
    try:
        day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
        day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))

        real_time_LMP_rev = real_time_LMP * kW_load/(12)
        real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))

        mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
        mining_rev = float("{0:.7f}".format(mining_rev))

        return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev

    except Exception as e:
        print("Error:  " + str(e))

def main():
```

```
    try:
        block_height = get_block_height()
        print("Current Block Height:          " + str(block_height))

        BTC_price = get_BTC_price()
        BTC_price = float("{0:.2f}".format(BTC_price))
        print("Current BTC Price:             " + str(BTC_price))

        diff = get_network_difficulty()
        print("Network difficulty:            " + str(diff))

        hashrate = get_network_hashrate(diff)
        print("Estimated Network Hashrate:    " + str(hashrate))

        day_ahead_LMP = get_day_ahead_LMP(location)
        day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
        print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))

        real_time_LMP = get_real_time_LMP(location)
        real_time_LMP = float("{0:.7f}".format(real_time_LMP))
        print("Real Time LMP ($/kWh):         " + str(real_time_LMP))

        breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
        kW_load)
        breakeven = float("{0:.7f}".format(breakeven))
        print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
        print("_____________________________________________________________")

        profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)

        day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
        hashrate, real_time_LMP, day_ahead_LMP)

        if(day_ahead_LMP >= breakeven):
            realized_rev = day_ahead_LMP_rev
        elif(real_time_LMP >= breakeven):
            realized_rev = real_time_LMP_rev
        else:
            realized_rev = mining_rev

        conn = sqlite3.connect(db_file)

        conn.execute("CREATE TABLE IF NOT EXISTS arb_main_EXELON10_11 (datetime,
        block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev)")

        conn.execute("INSERT INTO arb_main_EXELON10_11 (datetime, block_height,"
                     "network_diff, est_network_hashrate, BTC_price, day_ahead_LMP,
                     real_time_LMP, breakeven_mining_cost, day_ahead_LMP_rev,
                     real_time_LMP_rev, mining_rev, realized_rev) "
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
                     (str(datetime.datetime.now()), block_height,
                      diff, hashrate, BTC_price, day_ahead_LMP, real_time_LMP,
                      breakeven, day_ahead_LMP_rev, real_time_LMP_rev, mining_rev,
                      realized_rev))
        conn.commit()

    except Exception as e:
        print("Error:  " + str(e))

while True:
    main()
    time.sleep(300)
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging, config
from pathlib import Path
from LMP_csv_import import get_real_time_LMP
from DA_LMP_import import get_day_ahead_LMP
from toggle_relay import *
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
from email_alert import *
from bitcoin.rpc import RawProxy

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 31000000 * 272
kW_load = 1.8 * 272
location = 'AEC'
block_reward = 12.5

def get_BTC_price():
    try:
        price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
        price_dict = price.json()
        BTC_price = price_dict['data']['amount']
        BTC_price = float(BTC_price)
        return BTC_price

    except Exception as e:
        print("Error:  " + str(e))

def get_network_difficulty():
    try:
        p = RawProxy()
        diff = float(p.getdifficulty())

        return diff

    except Exception as e:
        print("Error:  " + str(e))

def get_block_height():
    try:
        p = RawProxy()
        info = p.getblockchaininfo()
        block_height = (info['blocks'])

        return block_height

    except Exception as e:
            print("Error:  " + str(e))

def get_network_hashrate(diff):
    try:
        network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
        hashrate_dict = network_hashrate.json()
        # print(hashrate_dict)
        hashrate = hashrate_dict['data']['hashrate']
        hashrate = float(hashrate) / 1000000
        # hashrate = int(hashrate)
        # print("Called network hashrate: " + str(hashrate))
        return hashrate

    except Exception as e:
        print("Error:  " + str(e))
```

```
65
66  def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
67      try:
68          breakeven = ((miner_hashrate / hashrate) * (block_reward*144) * (BTC_price)) /
            (kW_load * 24)
69          #print(breakeven)
70          return breakeven
71
72      except Exception as e:
73          print("Error:  " + str(e))
74
75  def pdu1_all_on():
76      try:
77          from pymodbus.client.sync import ModbusTcpClient as ModbusClient
78          client = ModbusClient(config.pdu1_ip, config.pdu1_port)
79          client.write_coil(0, False)
80          client.write_coil(1, False)
81          client.write_coil(2, False)
82          client.write_coil(3, False)
83          client.write_coil(4, False)
84          client.write_coil(5, False)
85          client.write_coil(6, False)
86          client.write_coil(7, False)
87          client.write_coil(8, False)
88          client.write_coil(9, False)
89          client.write_coil(10, False)
90          client.write_coil(11, False)
91          client.write_coil(12, False)
92          client.write_coil(13, False)
93          client.write_coil(14, False)
94          client.write_coil(15, False)
95          client.write_coil(16, False)
96          client.write_coil(17, False)
97          client.write_coil(18, False)
98          client.write_coil(19, False)
99          client.write_coil(20, False)
100         client.write_coil(21, False)
101         client.write_coil(22, False)
102         client.write_coil(23, False)
103
104         client.close()
105
106     except Exception as e:
107         print("Error:  " + str(e))
108
109 def pdu1_all_off():
110     try:
111         from pymodbus.client.sync import ModbusTcpClient as ModbusClient
112         client = ModbusClient(config.pdu1_ip, config.pdu1_port)
113         client.write_coil(0, True)
114         client.write_coil(1, True)
115         client.write_coil(2, True)
116         client.write_coil(3, True)
117         client.write_coil(4, True)
118         client.write_coil(5, True)
119         client.write_coil(6, True)
120         client.write_coil(7, True)
121         client.write_coil(8, True)
122         client.write_coil(9, True)
123         client.write_coil(10, True)
124         client.write_coil(11, True)
125         client.write_coil(12, True)
126         client.write_coil(13, True)
127         client.write_coil(14, True)
128         client.write_coil(15, True)
```

```
129             client.write_coil(16, True)
130             client.write_coil(17, True)
131             client.write_coil(18, True)
132             client.write_coil(19, True)
133             client.write_coil(20, True)
134             client.write_coil(21, True)
135             client.write_coil(22, True)
136             client.write_coil(23, True)
137
138             client.close()
139
140         except Exception as e:
141             print("Error:  " + str(e))
142
143     def profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven):
144         try:
145             if(day_ahead_LMP >= breakeven):
146                 #pdu1_all_off()
147                 email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
148                 return
149             elif(real_time_LMP > breakeven):
150                 #pdu1_all_off()
151                 email_alert_miners_off(real_time_LMP, breakeven, day_ahead_LMP)
152                 return
153             #else:
154                 #pdu1_all_on()
155
156         except Exception as e:
157             print("Error:  " + str(e))
158
159     def get_profit(BTC_price, hashrate, real_time_LMP, day_ahead_LMP):
160         try:
161             day_ahead_LMP_rev = day_ahead_LMP * kW_load/(12)
162             day_ahead_LMP_rev = float("{0:.7f}".format(day_ahead_LMP_rev))
163
164             real_time_LMP_rev = real_time_LMP * kW_load/(12)
165             real_time_LMP_rev = float("{0:.7f}".format(real_time_LMP_rev))
166
167             mining_rev = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
168             mining_rev = float("{0:.7f}".format(mining_rev))
169
170             return day_ahead_LMP_rev, real_time_LMP_rev, mining_rev
171
172         except Exception as e:
173             print("Error:  " + str(e))
174
175     def main():
176         try:
177             block_height = get_block_height()
178             print("Current Block Height:          " + str(block_height))
179
180             BTC_price = get_BTC_price()
181             BTC_price = float("{0:.2f}".format(BTC_price))
182             print("Current BTC Price:             " + str(BTC_price))
183
184             diff = get_network_difficulty()
185             print("Network difficulty:            " + str(diff))
186
187             hashrate = get_network_hashrate(diff)
188             print("Estimated Network Hashrate:    " + str(hashrate))
189
190             day_ahead_LMP = get_day_ahead_LMP(location)
191             day_ahead_LMP = float("{0:.7f}".format(day_ahead_LMP))
192             print("Day Ahead LMP ($/kWH):         " + str(day_ahead_LMP))
193
```

```
194             real_time_LMP = get_real_time_LMP(location)
195             real_time_LMP = float("{0:.7f}".format(real_time_LMP))
196             print("Real Time LMP ($/kWh):          " + str(real_time_LMP))
197
198             breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
                kW_load)
199             breakeven = float("{0:.7f}".format(breakeven))
200             print("Breakeven Mining Cost ($/kWh): " + str(breakeven))
201             print("_____________________________________________________________")
202
203             #profit_comp_controller(day_ahead_LMP, real_time_LMP, breakeven)
204
205             day_ahead_LMP_rev, real_time_LMP_rev, mining_rev = get_profit(BTC_price,
                hashrate, real_time_LMP, day_ahead_LMP)
206
207             if(day_ahead_LMP >= breakeven):
208                 realized_rev = day_ahead_LMP_rev
209             elif(real_time_LMP >= breakeven):
210                 realized_rev = real_time_LMP_rev
211             else:
212                 realized_rev = mining_rev
213
214             conn = sqlite3.connect(db_file)
215             conn.execute("INSERT INTO denis_logic_test_newgen (datetime, day_ahead_LMP,
                block_height, BTC_price, "
216                          "network_diff, est_network_hashrate, real_time_LMP,
                           breakeven_mining_cost, day_ahead_LMP_rev, real_time_LMP_rev,
                           mining_rev, realized_rev) "
217                          "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);",
218                          (str(datetime.datetime.now()), day_ahead_LMP, block_height,
                           BTC_price,
219                           diff, hashrate, real_time_LMP, breakeven, day_ahead_LMP_rev,
                           real_time_LMP_rev, mining_rev, realized_rev))
220             conn.commit()
221
222         except Exception as e:
223             print("Error:  " + str(e))
224
225     while True:
226         main()
227         time.sleep(300)
228
```

```python
#!/usr/bin/env python3
import smtplib, ssl, socket, json, requests, urllib3, datetime, sqlite3, os, time, logging
from pathlib import Path
from LMP_csv_import import get_realtime_LMP

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

folder = Path("C:/Users/Admin")
db_file = str(folder / "test_10.sqlite")
miner_hashrate = 14300000*272
kW_load = 1.3*272*1.05
location = 'AEC'
block_reward = 144

def get_BTC_price():
    price = requests.get('https://api.coinbase.com/v2/prices/spot?currency=USD')
    price_dict = price.json()
    BTC_price = price_dict['data']['amount']
    BTC_price = float(BTC_price)
    return BTC_price

def get_network_difficulty():
    network_difficulty = requests.get('https://blockexplorer.com/api/status?q=getDifficulty')
    diff_dict = network_difficulty.json()
    diff = diff_dict['difficulty']
    return diff

def get_block_height():
    block_height =requests.get('https://blockexplorer.com/api/status?q=getBlockCount')
    bh_dict = block_height.json()

    #print(bh_dict)
    block_height = bh_dict['info']['blocks']
    block_height = int(block_height)
    print(str(block_height))
    return block_height

def get_network_hashrate(diff):
    network_hashrate = requests.get("https://chain.so/api/v2/get_info/BTC")
    hashrate_dict = network_hashrate.json()
    #print(hashrate_dict)
    hashrate = hashrate_dict['data']['hashrate']
    hashrate = float(hashrate)/1000000
    #hashrate = int(hashrate)
    #print("Called network hashrate: " + str(hashrate))

    expected_blocks = 144
    blocks_found = 113
    est_hashrate = (blocks_found / expected_blocks * diff * 2 ** 32 / 600)
    #print("Estimated network hashrate: " + str(est_hashrate))
    return hashrate

def get_profit(BTC_price, hashrate, lmp_price):
    gross_profit_USD = ((miner_hashrate / hashrate) * 6.25 * BTC_price * .985)
    print("5m gross USD profit:   $" + str(gross_profit_USD))

    power_cost_USD = kW_load*lmp_price/6/1000
    print("5m electricity cost:   $" + str(power_cost_USD))

    net_profit_USD = gross_profit_USD - power_cost_USD
    print("5m net USD profi/loss: $" + str(net_profit_USD))

    net_profit_BTC = net_profit_USD/BTC_price
```

```
64      print("5m net BTC profit:     " + str(net_profit_BTC))
65
66      return gross_profit_USD, power_cost_USD, net_profit_USD, net_profit_BTC
67
68  def get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price, kW_load):
69      breakeven = ((miner_hashrate/hashrate)*(block_reward*144)*(BTC_price))/(kW_load*24)
70      print(breakeven)
71      return breakeven
72
73  def main():
74      try:
75
76          print(str(datetime.datetime.now()))
77          block_height = get_block_height()
78          BTC_price = get_BTC_price()
79          diff = get_network_difficulty()
80          hashrate = get_network_hashrate(diff)
81          lmp_price = get_realtime_LMP(location)
82          breakeven = get_breakeven_USD_per_kWh(miner_hashrate, hashrate, BTC_price,
            kW_load)
83          gp_USD, pc_USD, np_USD, np_BTC = get_profit(BTC_price=get_BTC_price(),
            hashrate=get_network_hashrate(diff),
84                                                      lmp_price=get_realtime_LMP(location))
85
86          conn = sqlite3.connect(db_file)
87          conn.execute("INSERT INTO lmp_model (date_time, BTC_USD_price,
            BTC_network_hashrate, real_time_LMP_mwh, "
88                       "real_time_LMP_kwh, gross_profit_USD_5m, power_cost_USD_5m,
                         net_profit_USD_5m, net_profit_BTC_5m, block_height) "
89                       "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
                         ?);",(str(datetime.datetime.now()), BTC_price, hashrate, lmp_price,
90                                                              (lmp_price/1000), gp_USD,
                                                                pc_USD, np_USD, np_BTC,
                                                                block_height))
91          conn.commit()
92
93      except Exception as e:
94              print("Error:  " + str(e))
95
96  while True:
97      main()
98      time.sleep(300)
99
```

Exhibit CC

REDACTED IN ITS ENTIRETY

Exhibit DD

REDACTED IN ITS ENTIRETY